Demystifying cluster-based fault-tolerant Firewalls

Pablo Neira, L. Gasca, Laurent Lefèvre

Demystifying cluster-based fault-tolerant Firewalls

P. Neira, R.M. Gasca
QUIVIR Research Group
Department of Languages and Systems
ETS Ingeniería Informatica
Avda. Reina Mercedes, s/n
41012 SEVILLA - Spain
{pneira|gasca}@us.es

L. Lefèvre
INRIA RESO - Universite of Lyon
LIP Laboratory
(UMR CNRS, INRIA, ENS, UCB)
Ecole Normale Supérieure de Lyon
46 allée d'Italie
69364 LYON 07 - France
laurent.lefevre@inria.fr

***Abstract*—Firewalls are perimeter security solutions that are useful to address the unwanted traffic problem. However, firewalls cover network security aspects at the cost of introducing more problems in terms of network performance, availability and complexity that also need to be appropriately addressed. This work surveys existing cluster-based fault-tolerant firewall architectures and it discusses their trade-offs in performance, availability and complexity. We present preliminary evaluation of these architectures and we also discuss the need for state-replication in stateful firewall clusters.**

I. Introduction

In the early days of the Internet, the term *firewall* was used to refer to stateless packet filters. These packet filters control packet traversal across different network segments according to a given filtering policy. The filtering policy is expressed in an Access Control List (ACL), which is composed of rules. Each rule uses a vendor-specific low level language that contains packet selectors; like source and destination address, port source and destination, protocol type, and other fields available in the network packet header.

At some point, stateless packet filters proved to be uneffective against certain attacks like some port probing techniques, e.g. IP ID Idle and TCP-flag scans [1]; and other sorts of intentional or unintentional out-of-specification protocol misbehaviours. The main problem of stateless firewalls is that the filtering decision is based on individual packets, thus, the global evolution of the communication flow is ignored. Moreover, stateless packet filters have problems to filter multi-session application protocols like FTP or SIP. These protocols use different flows for control and data sessions, and packet filters cannot relate the flows since they operate from the network and transport layers.

In order to overcome these limitations, packet filter capabilities were extended to become stateful. Stateful firewalls [2] perform conformance checkings upon the filtered network traffic. Basically, they enforce that the communications between two peers evolve according to the protocol specification. Stateful firewalls keep track of the filtering by storing the current state of a flow, and they allow a set of possible transitions from a given state. Thus, the firewall administrator can modify the ACL to perform some action on the packets that trigger invalid state transitions, like logging and dropping them. Another interesting aspect is that stateful firewalls also usually support application layer inspection to successfully filter multi-session application protocols[1].

Although, in general, firewalls cover valuable security concerns, they also introduce more problems in terms of network complexity, performance and availability. In this work, we survey existing fault-tolerant firewall solutions, we discuss their trade-offs in performance, availability and complexity; we discuss the need for state-replication, and we present preliminary evaluation.

This article is organized as follows: in section II, we discuss challenges associated to firewalls in terms of network performance, availability and complexity. Then, section III presents common cluster-based firewall architectures, while section IV focuses on specific aspects of the need for state-replication in cluster-based stateful firewalls. The experimental evaluation is exposed in section V. Section VI lists related works and existing solutions; and section VII concludes this paper and presents some future works.

II. Firewall challenges: network performance, availability and complexity

Firewalls introduce a single point through which all traffic flows have to pass. This is a problem for network designers in terms of performance, since firewalls may increase latency and reduce bandwidth throughput; availability, since a failure may isolate the protected network and reduce the mean up time; and complexity, since firewalls are new network nodes that have to be appropriately configured and administered.

Performance is a major issue since firewalls may become a bottleneck in the network schema. As each packet that goes across has to match an ACL rule to go through, the matching approach must scale according to the number of rules. An interesting amount of research work addressing the packet matching problem and ACL design have been done since the late '90s [3]. Although scalable packet matching

[1]Some product refers to this ability as deep packet inspection.

algorithms and appropriate ACL design are important to improve performance, they are not covered by this work since we mainly focus on improving performance by means of clustering techniques.

With regards to availability, firewalls are single points of failures. Thus, a failure in the firewall leads to the isolation of the different connected network segments. Traditionally, redundancy has been used to overcome this potential limitation in packet filters. There are several different redundancy-based architectural designs for firewalls that we detail in this article. Although not covered in this work, link-level redundancy must also be considered to achieve fault-tolerant networks since network switches are also single points of failures. This problem can be addressed with link aggregation (IEEE 802.1AX-2008, formerly IEEE 802.3ad). Redundant ISP links are also important to achieve fault-tolerance. However, this is not in the scope of this article either.

Complexity is another issue that network designers have to face. Architecting highly available and performance firewall solutions incurs in some extra complexity in the network design. Firewalls are also inherently complex solutions that must be appropriately configured to provide an acceptable improvement in terms of security. Moreover, considering that attacks may also come from insiders, and depending on the network structure, a distributed firewall's architecture may be required to provide a good level of security, thus, more than one firewall may be required.

Therefore, according to these three parameters: performance, availability and complexity, the network designer's target must be to select the appropriate architecture to obtain good performance and availability, while trying to keep reduced complexity, at the same time, to simplify maintainance. However, as we expose in this work, network designers have to adopt a compromise solution since improving availability increases complexity and it may reduce performance.

III. Fault-Tolerant Cluster-based firewalls

There are different fault-tolerant and performance firewall architectures with different complexity degrees. In this section, we detail the most used. As we will expose, the common denominator of the following cluster architectures is that stateless firewalls are not problematic, from the availability perspective, whereas stateful firewalls are, for different reasons. In general, stateful firewalls require some state-replication solutions to fully support fault-tolerant firewall architectures. The need for state-replication also deserves special attention, as we do in this work, since it also has impact in the architecture in terms of performance and availability.

A. The Primary-Backup approach

The Primary-Backup firewall approach consists of having two or more firewalls that compose a cluster in which one acts as primary, and others act as backup (See Fig. 1). In this setup, a redundancy protocol like IETF's Virtual Router Redundancy Protocol (VRRP), OpenBSD's Common Address Redundancy Protocol (CARP) or Cisco's Hot Standby Router Protocol (HSRP) is used to ensure that at least one firewall is acting as primary at all times. These redundancy protocols are basically based on heartbeats to detect failures. Heartbeats are token messages that firewalls send periodically to each other if they are working. If the primary stops sending heartbeats, the redundancy protocol assumes that the primary is not available. Thus, it selects a new firewall among the backups to become the new primary. The primary owns the so-called virtual IPs (VIPs), which are the IP addresses used by peers as a default gateway. These VIPs are assigned to the primary firewall.

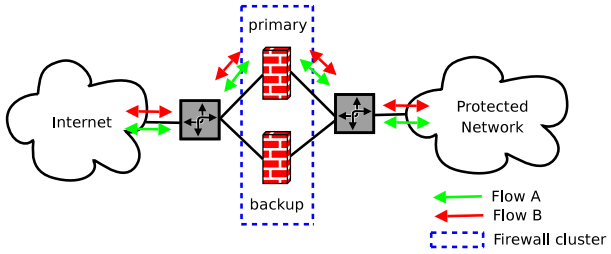

Fig. 1. An example of the Primary-Backup firewall architecture

This approach works fine in practise for stateless firewalls. However, if stateful firewalls are used, it leads to the breakage of established communications during the recovery. This problem is due to the absence of the state information in the backup firewalls. Some vendors provide a pickup facility to allow recovering flows from the middle, however, this mechanism reduces the security that stateful firewalls provide as it relaxes the conformance-checking done during the recovery.

From the resource use perspective, the Primary-Backup approach, although useful in terms of availability, is wasteful since the backup firewalls are idle, waiting for the primary firewall to fail. Therefore, this architecture does not provide any advantage in terms of performance, and every extra firewall node belonging to the cluster increases network complexity.

B. Multi-primary multi-path firewall cluster

A common practise to improve the resource use of the Primary-Backup approach has been multi-path routing. For simplicity, assume a cluster node composed of two firewalls, $C = \{fw_1, fw_2\}$, where the firewall fw_1 owns the VIP_1, which is the VIP that peers in the protected network use as default gateway, and the other firewall fw_2 owns VIP_2, which is the VIP that peers in the Internet use to route traffic to the peers that are in the protected network. Thus, fw_1 filters traffic leaving the protected network and fw_2 filters the traffic going to the protected network so that packets follows different paths according to their direction. With regards to availability, as exposed in the Primary-Backup approach, a redundancy protocol can be used to detect firewall failures. Thus, if the firewall fw_2 fails, the firewall fw_1 can recover the VIP_1 and filter both outgoing and incoming traffic until fw_2 is repaired.

This architecture improves availability and performance but it is only suitable for stateless firewalls. Note that, if stateful firewalls are used, some sort of state-replication solution would

be required to propagate state-changes between the firewalls. However, the propagation of state-changes would need to be synchronous, otherwise packets may race with the state propagation. Synchronous state-replication means that each packet would have to wait until one state-change is successfully propagated to the backup firewalls. As a result, this approach would dramatically reduce performance. Therefore, this renders state-replication unfeasible in this architecture.

C. Multi-primary firewall cluster sandwich

Another popular architecture is the *firewall sandwich*. In this architecture, the firewall cluster is placed between two load-balancers, one on each side. Thus, the load-balancers distribute the flows between the firewalls. This architecture moves the single point of failure problem from the firewalls to the load-balancers. Thus, we require redundant load-balancers and the use of a redundancy protocol to detect failures. Moreover, we also require some state-replication solution to ensure that the flow-based distribution remains the same across failures.

The main drawback of this architecture is the extra complexity added by the use of load-balancers (up to four, two on each side, assuming redundancy). Moreover, the load-balancers may become a bottleneck if the flow distribution does not scale accordingly to the number of firewall nodes that are used in the firewall cluster.

D. Multi-primary hash-based stateful firewall-clusters

The main feature of this architecture is that firewalls share the workload without the need of load-balancers. Thus, it reduces the complexity of the previous architecture. In this approach, we assume that all the firewalls in the cluster are clones: they all use the same IPs, and they all receive the same traffic at the same time. Then, a hash-based approach selects only one firewall to filter a given flow. Thus, the flows are distributed between the stateful firewalls (See Fig. 2).

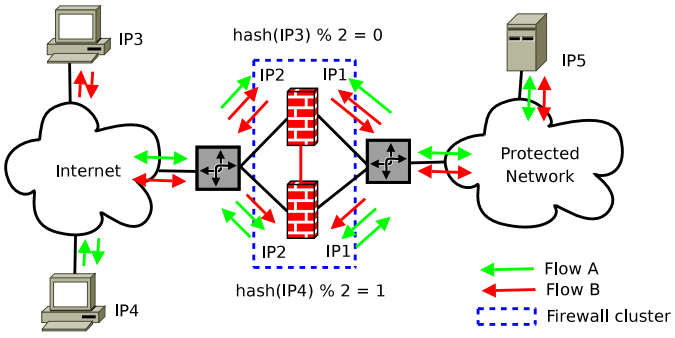

Fig. 2. An example of the hash-based multi-primary with two flows

In this architecture, each firewall has a unique id-number $id(fw_x)$ whose value can be $\{0, ..., n-1\}$ where n is the number of firewalls in the cluster. The id-number is manually set by the administrator. We assume that every flow F_j is represented by means of one flow-tuple $T_j = \{Address_{SRC}, Address_{DST}, Port_{SRC}, Port_{DST}, Protocol\}$. If the packet P_i that is part of the flow F_j arrives to the firewall fw_x, then a flow-tuple T_j is associated to the packet. Then, if $hash(Address_{SRC}(T_j))\ modulo\ n$ equals $id(fw_x)$, then this packet P_i must be filtered by the firewall fw_x. Thus, the flow distribution is achieved. Note that the real packet source is not used in the hashing, instead, the source address of the flow-tuple T_j is used. Therefore, the same value is used to calculate the hash for packets going in both directions to avoid multi-path filtering that breaks stateful filtering.

To make all firewalls receive the same packets, you can use a network-hub or configure your network-switch to act as a network-hub using some port mirroring feature. However, not all network-switches allow this configuration. In that case, a multicast MAC address (01:00:5e) is commonly used by all the firewalls. Thus, if the network-switch associates switch-ports and MAC addresses, the network-switch floods the same packets to the switch-ports which have the same multicast MAC address. Although this technique is commonly used in existing firewall products, it is a violation of the standard RFC 1812 since a network node must not use a multicast MAC address in an ARP reply message. A possible solution for this would be that network-switch vendors start supporting VRRP reserved MAC address space as multicast MAC addresses. However, that seems not to have happened so far.

Looking at this approach from the fault-tolerance perspective, if a firewall fails, a subset of flows would not be appropriately filtered. This problem can be avoided if each firewall has a mask of id-numbers that the firewall is responsible for. Thus, firewalls may filter flows for more than one id-number temporarily until the real owner of the id-number becomes active again. However, if stateful firewalls are used, the recovery of the established flows is not possible as exposed in the Primary-Backup approach. Therefore, state-replication would be required.

In conclusion, this architecture reduces the complexity with regards to the firewall sandwich solution as it does not need load-balancers. With regards to performance, some results are provided in the evaluation section of this work.

IV. Cluster-based fault-tolerant stateful firewalls: the need for state-replication

In cluster-based stateful firewalls, state-replication becomes the key idea to solve possible breakages of established communications. Since the current flow-states are crucial to keep deploying the stateful filtering appropriately, the primary firewall has to propagate state-changes to the backup firewalls. Thus, if the primary firewall fails, the backup firewall selected to become the new primary can successfully recover the filtering.

In some situations, the breakage of established flows may be particularly disturbing, for example:

1) Multimedia applications breakage, eg. popular youtube-like Internet video and radio streaming disruptions.
2) Remote control utility breakage, eg. SSH connection closures.
3) The interruption of a big data transfer between two peers, eg. bulk downloads that cannot be resumed and

would need to be re-started.

4) Extra monetary cost for an organization, eg. if the VoIP communications are disrupted, the users would have to be re-called with the resulting possible extra cost.
5) Violation of a Service-Level Agreement, that may decrease the customer's confidence on a given service provider.

Although state-replication is useful to avoid any of these disturbing events, it does not come costless. Since the firewalls have to invest computational resources in the state-replication, network performance may be reduced.

At this point, the network designer has to select if it enables state-replication or not. This decision depends on how important availability and performance are in the network design of the cluster-based stateful firewall. However, it also depends on how well state-replication can improve availability. In the following section, we evaluate the need for state-replication in the different scenarios that we have surveyed in this work.

V. Evaluation

We have evaluated the exposed cluster-based firewall architectures in our testbed network. Our environment consists of four HP Proliant 145g2 with one AMD dual core 2.2GHz using 1 GEthernet network links. Two of the systems have been used as firewalls and the other two act as HTTP client and server respectively. We have also tuned the interrupt handling affinity of each CPU core. Thus, the network card's interrupts in the firewalls are always handled by the same CPU core. For the cluster sandwich architecture, we have also used one Dell PowerEdge R410 with two Intel Xeon quad core 2.6GHz as load-balancer. This system is significantly superior to ensure that the load-balancer saturates the two firewalls that compose the cluster.

We have selected Linux as operating system, since it is widely used in the firewall industry[2], and the *iptables* [4] firewalling tool, that is one of the most popular free software firewalling tools. To evaluate state-replication we have used the *conntrack-tools* [5]. This software package contains a userspace daemon that can be configured to propagate state-changes asynchronously between firewalls. We have summarized the results of the experimentations and the survey in table I.

The experiments that we have performed to evaluate the fault-tolerant firewall architectures depend on two performance metrics:

1) Session rate, that determines the number of flows per second that the cluster-based firewall architecture can filter. In order to obtain the maximum session rate that the solution can filter, we have to generate flows with no data transfers. Thus, the flows are set up and tear down almost inmediately. This is the worst case for state-replication since it generates the maximum number of state-changes per second.
2) Data rate, that talks about the amount of data per second that the solution can filter. This metric is expressed in bytes per second. To obtain the maximum data rate we have to generate flows that perform large data transfers. The maximum data rate provides the best case for state-replication since it generates the minimum number of state-changes per second.

To evaluate availability, we have used the average number of recovered flows and their statistic deviation. The failures have been triggered by unplugging the dedicated link, that is used to replicate state-changes, and one of the main links, that connects the firewalls with the HTTP client and server. We have selected the *keepalived* [6] high-availability manager.

We have used the *inject32* [7] and the *httpterm* [8] tools to generate HTTP traffic between the client and the server. Basically, the clients start TCP sessions with the server and they retrieve one object of variable size per flow. The firewalls perform stateful filtering and Network Address Translation (NAT).

A. Primary-Backup firewall cluster

In Fig. 3, we have represented in the top part the relation between sessions/s (left y-axis) and throughput (right y-axis), which is an inverse relationship. To obtain the maximum throughput, we have used big objects (x-axis), whereas to obtain the maximum session/s we have used very small objects. We consider that small objects are those that are below the average web object size [10].

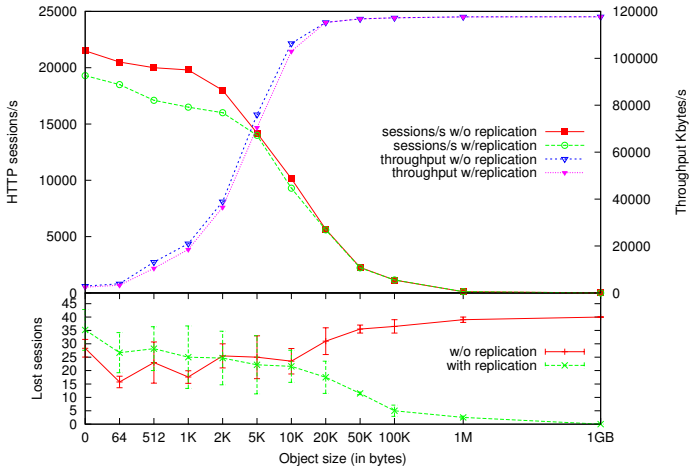

Fig. 3. Primary-Backup firewall cluster

Note that the number of sessions/s is around 10-15% less with state-replication enabled for objects smaller than 5KB. However, as it is shown in the bottom part of the figure, the recovery results are slightly worse with state-replication than without it. So, contrary to what one would expect, state-replication does not always improve the recovery. This problem is due to the asynchronous nature of the state-replication.

[2]Several vendors use Linux as base system for their firewall products, this is the case of StoneGate, Checkpoint's Firewall-1, Fortinet, Astaro, Vyatta and some of the Nokia IPSO versions, although most of them seem to use their own firewall software, not *iptables*.

Basically, the state-change propagation has a maximum latency of around 200ms according to LatencyTOP [9] which is more than the time to set up and tear down short flows between the client and the server in our testbed scenario, that is 1.7ms for empty objects and 3.8ms for 10KB objects. Thus, the backup holds old states that confuse the stateful firewall during the recovery. Note that state-replication starts being useful for 20KB objects whose transfer takes 6.8ms. For 50KB objects, the time to retrieve the object is 16.9ms, 100KB objects need 34.4ms and 1MB objects take 344ms. Therefore, to enable state-replication in a scenario in which flow-states have a very short lifetime is not a good idea. In the particular case of HTTP servers that are protected by firewalls, an evaluation of the average object size of your web server [10] would help to select if state-replication needs to be enabled.

B. Multi-primary multi-path firewall cluster

We have performed a very simple performance evaluation of this setup since stateful firewalling cannot be supported without problems due to the asynchronous nature of the state-replication approach. Basically, this architecture allows us to saturate the maximum sessions/s that the HTTP server can handle, which is 36000 session/s, and it reaches the maximum link throughput of 1GB. In this case, the HTTP server becomes the bottleneck. Therefore, some web server cluster solution would be required to reach the maximum session/s that both firewalls can handle.

As we have exposed in this work, multi-primary multi-path firewall clusters can be only considered for stateless firewalls. Some network administrators initially consider that state-replication is the solution to support stateful firewalling in their asymmetric-path firewall cluster. This conception is wrong due to the asyncronous nature of the state-replication. Moreover, in the particular case of Linux, stateless firewalls do not support NAT in this solution since this feature depends on the stateful filtering capabilities.

C. Multi-primary firewall sandwich cluster

The results of the experimental evaluation of the firewall sandwich cluster are shown in Fig. 4.

With regards to the sessions/s, we reach the limit of 36000 sessions/s that the HTTP server can handle. The performance drop in terms of session/s is around 15% with state-replication. With regards to flow recovery, similar conclusions to the Primary-Backup scenario can be extracted from the results. This architecture provides around 65% more performance than the Primary-Backup setup but this improvement is limited due to the HTTP server.

D. Multi-primary hash-based stateful firewall-clusters

Fig. 5 shows that the session/s that this solution can reach is around 40% more than the Primary-Backup setup. The scalability of this solution is limited due to the interrupt handling. Basically, the packet fetching from the NIC needs to be done to calculate the hash on the packet. Thus, each firewall has to invest resources to fetch a packet that may be ignored

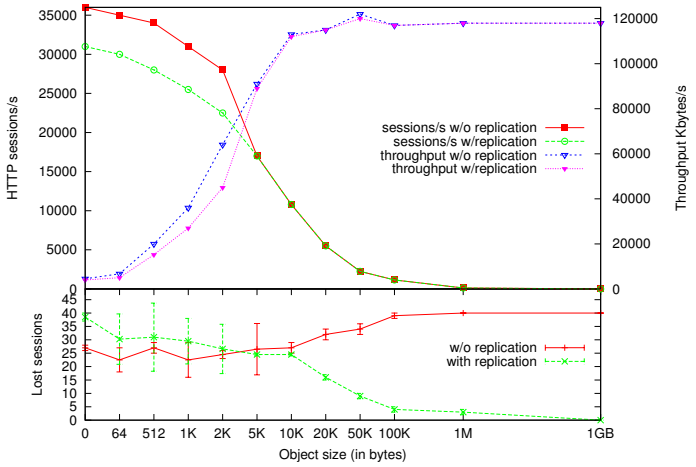

Fig. 4. Firewall sandwich cluster

if it does not belong to it. Nevertheless, this solution reduces network complexity since we do not require load-balancers. The bottom part of the figure shows that the recovered flows are slightly worse than the Primary-Backup and the Firewall sandwich cluster solutions. This occurs because the CPUs are mostly busy doing interrupt handling work. Note that this solution with state-replication enabled provides similar performance results than the Primary-Backup setup without state-replication.

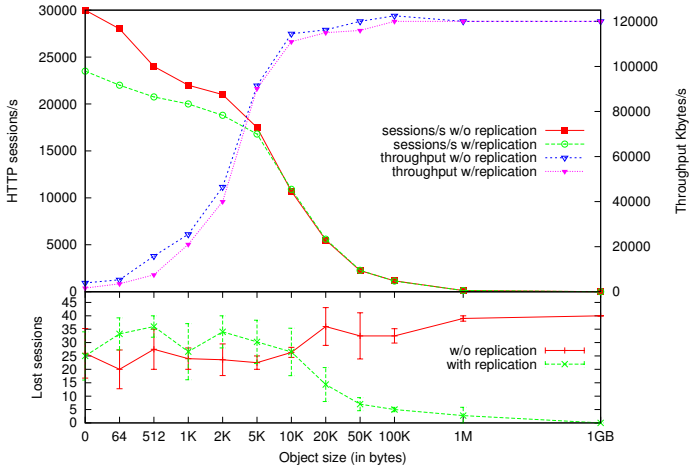

Fig. 5. Multi-primary hash-based firewall cluster

VI. Related works

There are several proprietary commercial solutions such as CheckPoint Firewall-1, StoneGate and Cisco PIX that offer highly available stateful firewall products that support the multi-primary hash-based cluster. However, as far as we know, there is only documentation on how to install and configure the firewall cluster. Unfortunately, we did not find any relevant information on their internal design and implementation.

Cluster Type	Availability	Performance	Complexity
Primary-Backup	With no state-replication, established flows are not recovered.	No improvement since backup firewalls are idle. Reduced if state-replication used.	Requires extra backup firewall and network switches.
Multi-Primary multipath	Not suitable for stateful firewalls.	No idle stateless firewalls in the cluster.	Requires extra firewall and network switches.
Multi-Primary sandwich	Move single point of failure from firewalls to load-balancers.	Load-balancers must be stateful and they must scale according to the number of firewalls.	Requires four load-balancers to be fault-tolerant.
Multi-Primary hash-based	With no state-replication, established flows are not recovered.	40-45% more per firewall added according to evaluation.	Requires extra firewall nodes, but no need for load-balancers.

TABLE I
SUMMARY OF FAULT-TOLERANT CLUSTER-BASED FIREWALL ARCHITECTURES

In the open-source community, the OpenBSD project provides a fault-tolerant solution for their stateful firewall, the so-called *pfsync* [11], which is a kernelspace state-replication that uses an unreliable protocol to propagate state-changes. There are several documents disecting some implementation details but its source code remains the main reference. The implementation of the state-replication in kernelspace can help to reduce the latency in the state-change propagation. However, bugs in the state-replication code may lead to availability problems[3].

There are also several open-source projects like Linux-HA [12], CARP [11] and Keepalived [6] that implements heartbeat-based redundancy protocols.

With regards to firewall architectures, Kopparapu's book [13] provides a good background on cluster-based firewall architectures. However, this book barely covers stateful firewalls and state-replication aspects, probably due to the fact that stateful firewalls started to proliferate at the same time that the book was released.

VII. FUTURE WORKS: TOWARDS PROACTIVE FAULT-TOLERANCE FOR CLUSTER-BASED STATEFUL FIREWALLS?

Commercial off-the-shelf platforms provide, more and more, integrated hardware error-detection facilities. These mechanisms allow correction of hardware errors in runtime. These include RAM memory and PCI bus transfer error detection and correction [14]. Intuitively, if a system has correctable errors, it will experience performance degradation. Moreover, correctable errors may become uncorrectable at some point, and the chances of experiencing a failure increases. Following this basis, it would be safe to proactively migrate states to a sane operational firewall before the major failure. Thus, this approach will avoid costly preventive state-replication. Although there are some research works in this direction nowadays, we will have to wait until a feasible pure proactive fault-tolerant solution for cluster-based enterprise-level firewalls becomes reality.

[3]Although this may seem a weak argument, in a recent re-design of *pfsync*, a bug in the state-replication that can generate data corruptions was discovered: http:undeadly.org/cgi?action=article&sid=20090301211402.

REFERENCES

[1] Gordon Lyon, *Nmap network scanning*. insecure.org. [Online]. Available: http://nmap.org/book/

[2] M. Gouda and A. Liu, "A model of stateful firewalls and its properties," *Proceedings of International Conference on Dependable Systems and Networks, 2005 (DSN)*, pp. 128–137, 28 June-1 July 2005.

[3] David E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Computing Surveys*, vol. 37, no. 3, pp. 238–275, 2005.

[4] Netfilter Core Team, "iptables: The free software firewalling tool for linux," http://iptables.netfilter.org.

[5] Pablo Neira Ayuso, "conntrack-tools: The netfilter's connection tracking userspace tools," http://conntrack-tools.netfilter.org.

[6] Alexandre Cassen, "Keepalived: Health checking for LVS & high availability," http://www.linuxvirtualserver.org.

[7] Willy Tarreau, "inject32: HTTP injection tool." [Online]. Available: http://1wt.eu/tools/inject/inject32.c

[8] ——, "httpterm: HTTP server benchmarking tool." [Online]. Available: http://1wt.eu/tools/httpterm/

[9] Arjan van de Ven, "LatencyTOP: measuring and fixing Linux latency," 2008. [Online]. Available: http://www.latencytop.org/

[10] Andrew B. King, *Website Optimization: Speed, Search Engine & Conversion Rate Secrets*. O'Reilly, 2008. [Online]. Available: http://www.websiteoptimization.com/speed/tweak/average-web-page/

[11] Ryan McBride, "Pfsync: Firewall Failover with pfsync and CARP," http://www.countersiege.com/doc/pfsync-carp/.

[12] A. Robertson, "Linux HA project." [Online]. Available: http://www.linux-ha.org

[13] Chandra Kopparapu, *Load-Balancing servers, firewalls and caches*. Wiley, 2002.

[14] "Linux EDAC (Error Detection and Correction) Project," http://bluesmoke.sourceforge.net.