Multiprimary support for the availability of cluster-based stateful firewalls

Pablo Neira Ayuso, Rafael M. Gasca, Laurent Lefevre
<pneira@us.es>, <gasca@us.es>, <lefevre@inria.fr>

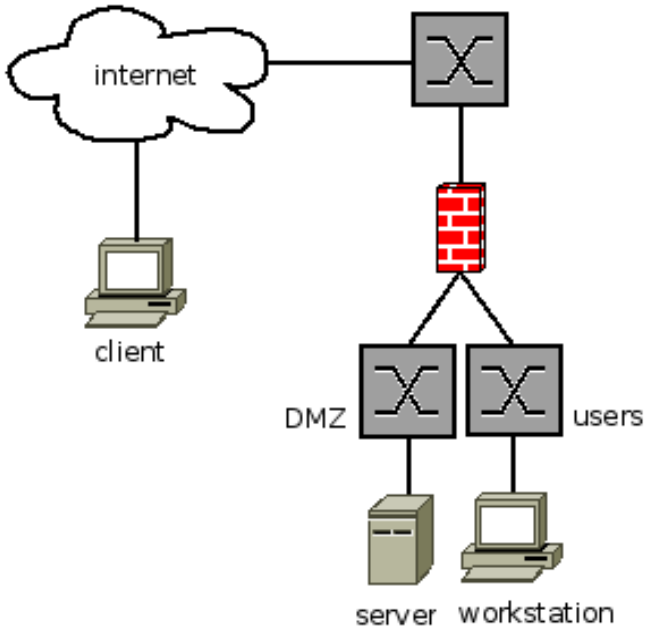
Quivir Research Group, University of Sevilla, INRIA/University of Lyon

Outline

- Short introduction:
  - Stateless and stateful firewalls. Fault-tolerant firewalls.
- Related works
- General description of our proposed solution: FT-FW
  - Architecture
  - Replication algorithm
  - Multiprimary support
- Evaluation
- Future works

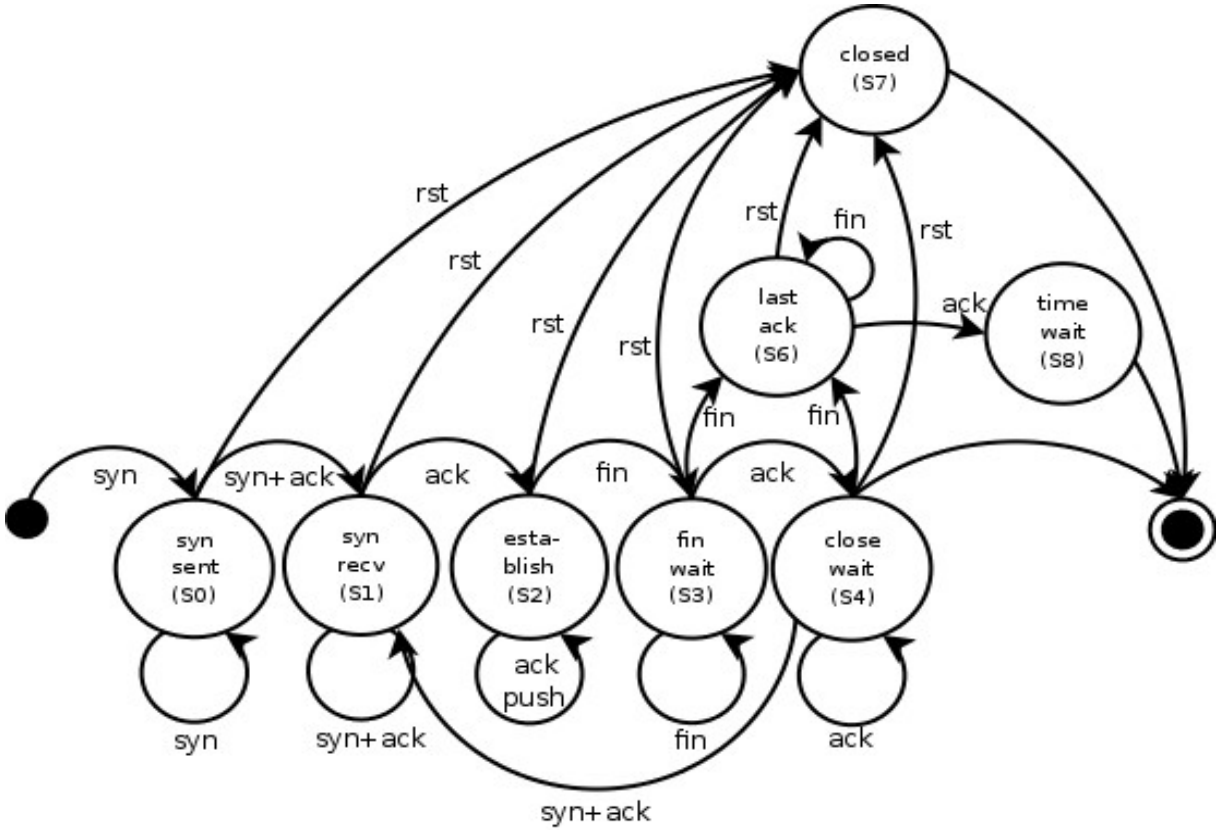
Firewalls: a very short introduction

- A Firewall is a network element that controls the traversal of packets across different network segments.
- Firewalls enforce an access control list (ACL). The ACL is composed of a list of linearly ordered filtering rules.
- If the firewall is ***stateless***, each rule contains a set of packet selectors (for matching purposes) and one action (eg. accept, drop).
- If the firewall is ***stateful***, it extends the stateless approach by performing conformance checkings upon the protocol that it filters.



Stateful firewalls: introduction

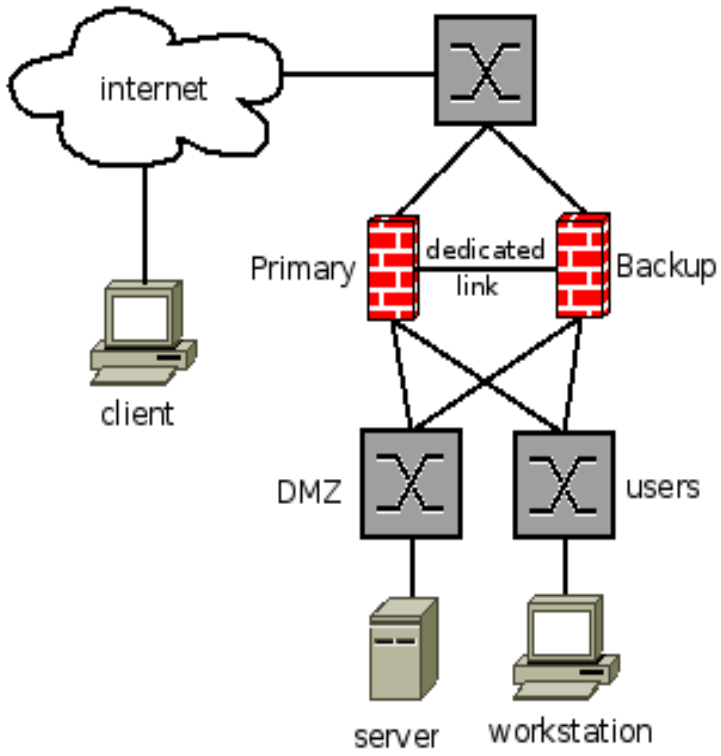
- ***Stateful*** firewalls enforce the correct evolution of a communication between two peers.
- The stateful firewall stores a set of variables to represent the current state of a flow F (tuple to identify the communication + state)
- Logic: when a packet P arrives:
  - is there any matching rule in the ACL for P?
  - does P triggers a valid state transition?



Example: simple TCP protocol state-machine

Firewalls: fault-tolerance

- The fault-tolerance problem: Firewalls introduce a **single point of failure.** For stateless firewalls, **system-level replication** is enough.
- However, this approach is insufficient for **stateful firewalls.** The backup firewall needs to know the current state of the variable.
- Replicas cannot one-copy equivalences in this case: we have to trade off between **flow durability** and performance.
- Primary-backup approach is wasteful in terms of resource use.



Firewalls: related works

- Many research has been done with regards to firewall during the last decade:
  - **Rule-set design:** Difficult task, it requires low-level languages.
  - **Rule-set consistency:** Rule-sets may disjoint (use of wildcards can result in ovelapping rules).
  - **Computational complexity:** Packet must be checked against a list of ordered rules. Algorithms to match packets that tradeoff with memory and CPU consumption.
  - **Distributed firewalls:** Scalable distributed design, ruleset delivery and, again, computational complexity problems.
  - **Fault-tolerance:** Firewalls introduce a single point of failure. Very few indeed wrt. firewalls.

Fault-tolerance: related works

- Many research has been done in the last three decades in the field of the application-level fault-tolerance:
  - Primary-Backup and State-machine replication.
  - Databases: Synchronous transactions to guarantee high degree of consistency (mostly eager replication).
  - Back-end servers: BTCP, FT-TCP, FT-WEB, etc.
  - Corba: FT-CORBA.
  - VoIP PBX
  - ...
- Stateful Firewall have different semantics that can be exploited.

Fault-tolerance firewalls: related works

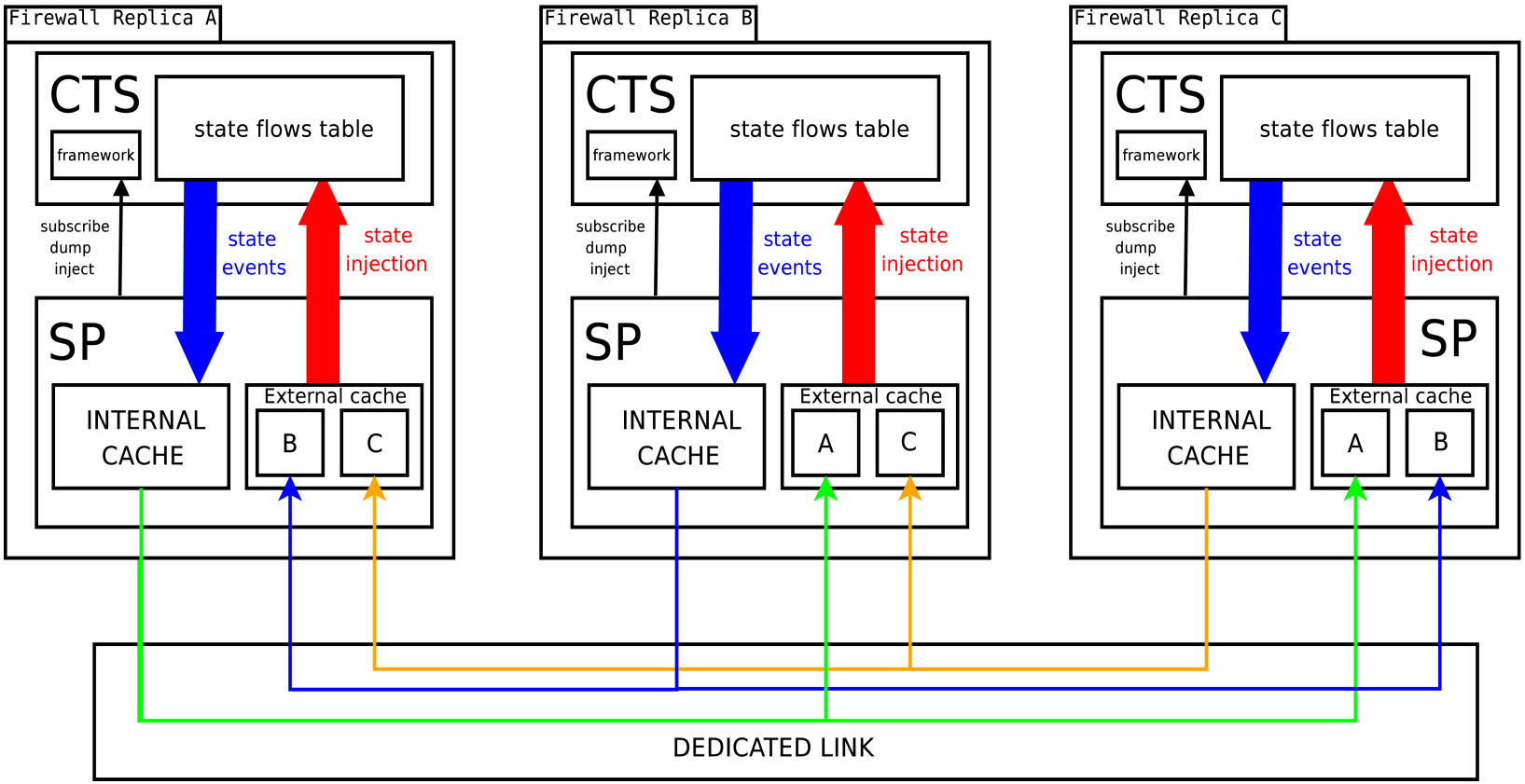
- Apart of the interesting openBSD effort, very few and nothing **open** in this area:
  - **Proprietary**: black box, only commercial papers (only commercial documentation available).
  - **OpenBSD**
    - Very simple in-kernel state replication and crash-only model.
    - Replication protocol: no sequence tracking at all, no message omission, reordering, duplication handling (only corruption is handled).

Enhancing fault-tolerance: FT-FW

- Reactive fault-tolerance at application level.
- Independent of the failure detection schema (eg. VRRP)
- **Transparency:** Negligible delay in client responses and quick recovery from failures. Asynchronous replication that trade-offs between flow durability and performance (in terms of bandwidth).
- **Simplicity:** Client transparent solution. The firewall requires minimal and non-intrusive extension.
- **Low cost:** Suitable for off-the shelf, software-based solution.
- **Multiprimary support:** load-sharing to avoid resource waste.
- Our approach aims to exploit the stateful firewall semantics.

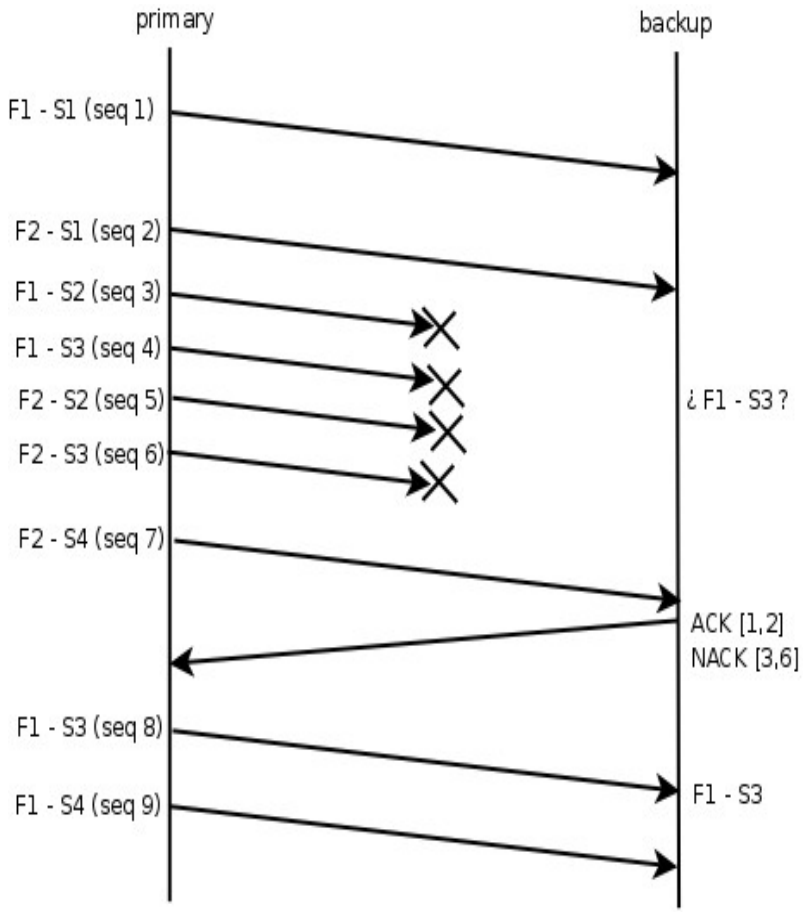
FT-FW architecture

- **Event-driven architecture**
- **CTS:** Connection Tracking System tracks connections and store states, we add extended it with a **framework** to inject states and receive state change events.
- **SP:** State proxy, replication daemon which interact with CTS. It stores two caches.



FT-FW replication protocol

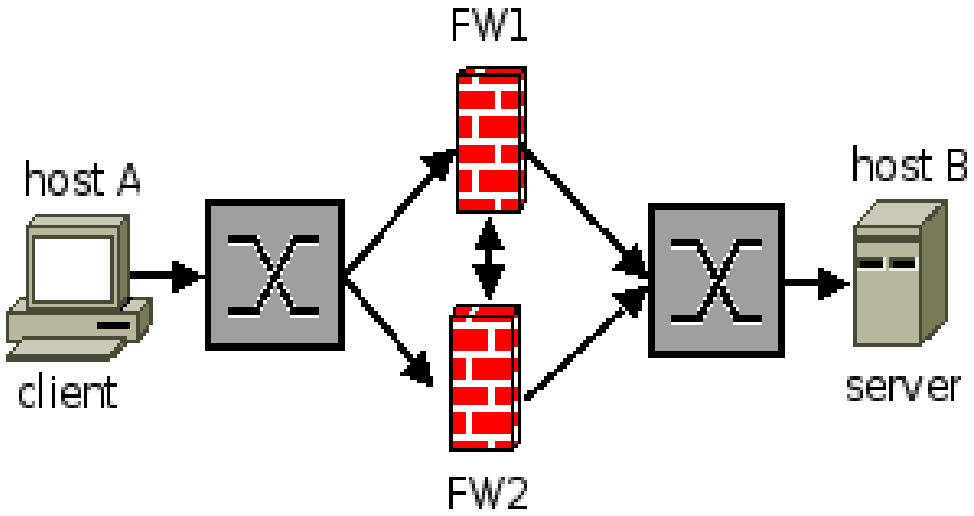
- We have to replicate states asynchronously: replicas are not one-copy equivalence.
- The **SP** is composed of two parts, the **sender** and the **receiver,** the replication protocol exploits the semantics of stateful firewalls:
  - The **sender** never stops sending messages
  - The **receiver** handles all messages (even those that are out of sequence).
  - The protocol reduces the number of retransmitted messages under message omission: It only resends the last state reached, not the whole state history.



FT-FW: multiprimary support

- We support two load-sharing approaches to remove the resource waste:
  - Static or client-based, ie. we set different gateways for clients via DHCP.
  - Dynamic or hash-based packet distribution between the firewall replicas: The packet P is seen by all the replica firewalls {FW_1, FW_2, ... FW_n} but only FW_k handles the packet.
    - Each node has an unique identifier (nid).
    - hash(src) % total_nodes & (1 << nid) == true ?
      If so, then handle the packet.
- Depending on the path routing, we support:
  - Symmetric path and the cache write back policy.
  - Asymmetric path and the cache write through policy.

Evaluation: testbed setup

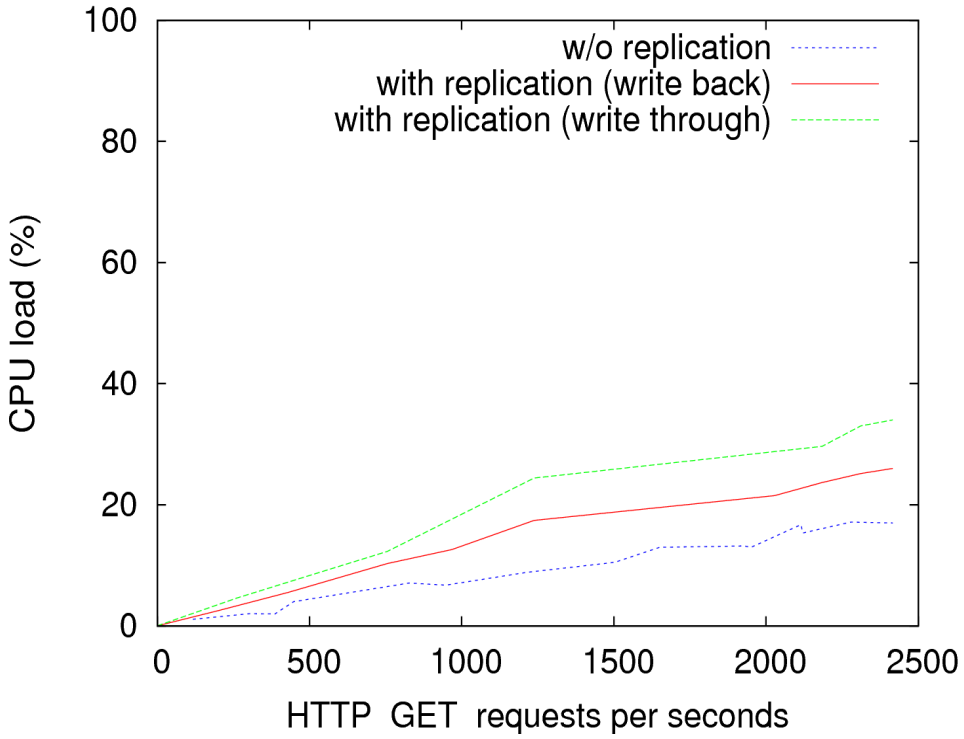


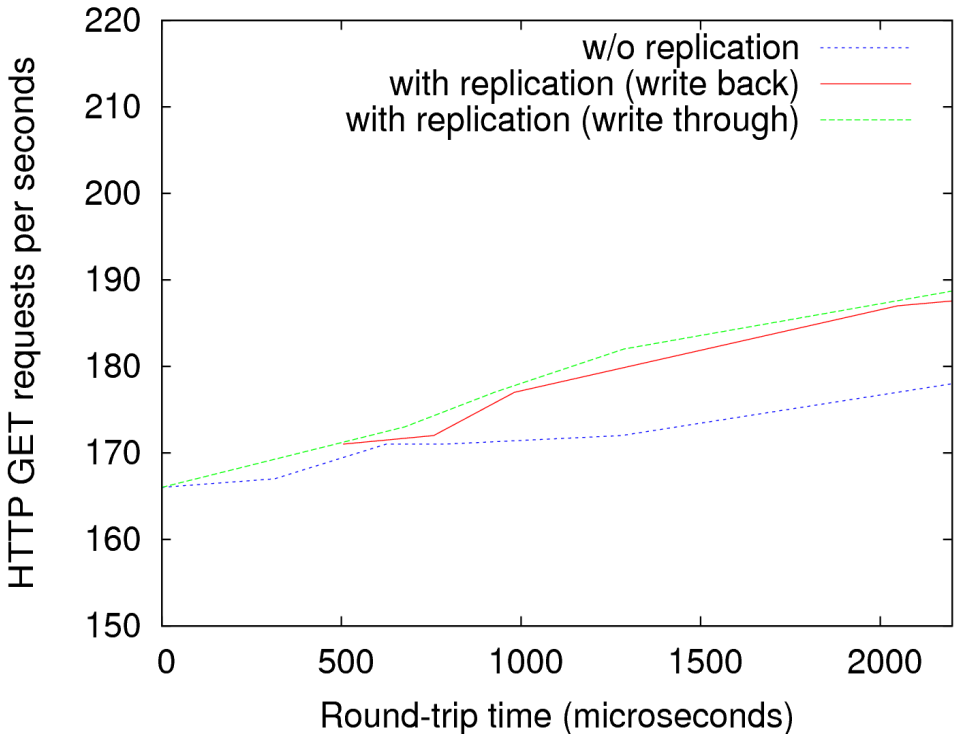
- Machine specs: HP Proliant 145g2, AMD Opteron2.2GHz, 1 GEthernet.
- conntrack-tools: free software (GPL) userspace daemon which implements the SP.

http://people.netfilter.org/pablo/conntrack-tools/

Evaluation: Replication MP

- **CPU consumption** during replication (1 TCP connection means 6 state changes): ~30% for 2500 HTTP GET connections per second.
- **Ping roundtrip:** The solution introduces a delay of ~4 milliseconds (negligible)





Future Works

- Reduce the overhead in the replication by means of hybridation: mixture of proactive and reactive fault-tolerance.
    - Exploit error detection and correction hardware and software facilities to enable proactive fault-tolerance.
    - Exploit the semantics of stateful firewalls to relax reactive replication – **trade-off between number of messages and flow durability.**
- Then, what if an error happens?
    - Was it detected by the error detection facilities? The proactive approach migrates the states to a sane firewall replica.
    - ... but if it was it not? The relaxed reactive replication should recover the filtering.
    - ... what if reactive cannot recover? Bad luck :-(
- Distributed firewalling in mobile wireless networks: the fault-tolerance problem is contained in the hand-over problem.

Conclusions

- This works presents the FT-FW architecture to enable fault-tolerant stateful firewalls from the multiprimary perspective.
- Many research works in terms of firewalling but no previous related, open research in the fault-tolerance field apart from the community-based OpenBSD effort.
- Replication requires extra computational resources, by means of proactive fault-tolerance we plan to reduce such consumption.
- Our replication protocol exploits the semantics of stateful firewalls.

Questions?

Thank you for your attention!