ABSTRACT:

This paper introduces several novel load-balancing algorithms for distributing Session Initiation Protocol (SIP) requests to a cluster of SIP servers. Our load balancer improves both throughput and response time versus a single node while exposing a single interface to external clients. We present the design, implementation, and evaluation of our system using a cluster of Intel x86 machines running Linux. We compare our algorithms to several well-known approaches and present scalability results for up to 10 nodes. Our best algorithm, Transaction Least-Work-Left (TLWL), achieves its performance by integrating several features: knowledge of the SIP protocol, dynamic estimates of back-end server load, distinguishing transactions from calls, recognizing variability in call length, and exploiting differences in processing costs for different SIP transactions. By combining these features, our algorithm provides finer-grained load balancing than standard approaches, resulting in throughput improvements of up to 24% and response-time improvements of up to two orders of magnitude. We present a detailed analysis of occupancy to show how our algorithms significantly reduce response time.
THE SESSION Initiation Protocol (SIP) is a general-purpose signaling protocol used to control various types of media sessions. SIP is a protocol of growing importance, with uses in Voice over IP (VoIP), instant messaging, IPTV, voice conferencing, and video conferencing. Wireless providers are standardizing on SIP as the basis for the IP Multimedia System (IMS) standard for the Third Generation Partnership Project (3GPP). Third-party VoIP providers use SIP (e.g., Vonage,
Gizmo), as do digital voice offerings from existing legacy telecommunications companies (telcos) (e.g., AT&T, Verizon) as well as their cable competitors (e.g., Comcast, Time-Warner).

- A frequent mechanism to scale a service is to use some form of a load-balancing dispatcher that distributes requests across a cluster of servers.

- Almost all research in this space has been in the context of either the Web (e.g., HTTP) or file service (e.g., NFS).

**DISADVANTAGE OF EXISTING SYSTEM:**

- Their load balancer is content unaware because it does not examine the contents of a request.

**PROPOSED SYSTEM:**

- SIP is a transaction-based protocol designed to establish and tear down media sessions, frequently referred to as calls.

- The session-oriented nature of SIP has important implications for load balancing.
Transactions corresponding to the same call must be routed to the same server; otherwise, the server will not recognize the call. Session-aware request assignment (SARA) is the process where a system assigns requests to servers such that sessions are properly recognized by that server, and subsequent requests corresponding to that same session are assigned to the same server.

We introduce new algorithms that outperform existing ones. Our work is relevant not just to SIP, but also for other systems where it is advantageous for the load balancer to maintain sessions in which requests corresponding to the same session are sent by the load balancer to the same server.

This paper makes the following contributions.

- We introduce the novel load-balancing algorithms CJSQ, TJSQ, and TLWL, described above, and implement them in a working load balancer for SIP server clusters. Our load balancer is implemented in software in user space by extending the OpenSER SIP proxy.

- We evaluate our algorithms in terms of throughput, response time, and scalability, comparing them to several standard “off-the-shelf” distribution policies.
such as round-robin or static hashing based on the SIP Call-ID. Our evaluation tests scalability up to 10 nodes.

• We show that two of our new algorithms, TLWL and TJSQ, scale better, provide higher throughputs, and exhibit lower response times than any of the other approaches we tested. The differences in response times are particularly significant.

For low to moderate workloads, TLWL and TJSQ provide response times for INVITE transactions that are an order of magnitude lower than that of any of the other approaches. Under high loads, the improvement increases to two orders of magnitude.

**ADVANTAGES OF PROPOSED SYSTEM:**

- Relatively less overhead
- Load balancing can be improved by combining SARA

These results show that our load balancer can effectively scale SIP server throughput and provide significantly lower response times without becoming a bottleneck. The dramatic response time reductions that we achieve with TLWL and
TJSQ suggest that these algorithms should be adapted for other applications, particularly when response time is crucial.

**MODULES**

- Session Initiation Protocol
- User agents
- Transaction-Least-Work-Left
- Load balancer

**MODULES DESCRIPTION:**

Session Initiation Protocol

SIP is a signaling protocol designed to establish, modify, and terminate media sessions between two or more parties. Several kinds of sessions can be used, including voice, text, and video, which are transported over a separate data-plane protocol. SIP does not allocate and manage network bandwidth as does a network resource reservation protocol. SIP messages traverse the SIP overlay network, routed by proxies, to find the eventual destinations. Once endpoints are found, communication is typically performed directly in a peer-to-peer fashion. This work
focuses on scaling the server, rather than the proxy. SIP can run over many protocols such as UDP, TCP, TLS, SCTP, IPv4.

**User agents**

A SIP uniquely identifies a SIP user. This layer of indirection enables features such as location-independence and mobility. SIP users employ end points known as user agents. These entities initiate and receive sessions. They can be either hardware or software. User agents are further decomposed into User Agent Clients (UAC) and User Agent Servers (UAS), depending on whether they act as a client in a transaction (UAC) or a server (UAS). Most call flows for SIP messages thus display how the UAC and UAS behave for that situation. SIP uses HTTP-like request/response transactions. A transaction consists of a request to perform a particular method and at least one response to that request. The transaction is only completed when a final response is received, not a provisional response.

**Transaction-Least-Work-Left**

The Transaction-Least-Work-Left (TLWL) algorithm addresses load balancing issue by assigning different weights to different transactions depending on their relative costs. Counters are maintained by the load balancer indicating the
weighted number of transactions assigned to each server. New calls are assigned to the server with the lowest counter. TLWL estimates server load based on the weighted number of transactions a server is currently handling. TLWL can be adapted to workloads with other transaction types by using different weights based on the overheads of the transaction types. In addition, the relative costs used for TLWL could be adaptively varied to improve performance.

**Load balancer**

The Receiver receives requests which are then parsed by the Parser. The Session Recognition module determines if the request corresponds to an already existing session by querying the Session State. If so, the request is forwarded to the server to which the session was previously assigned. If not, the Server Selection module assigns the new session to a server using TLWL algorithm. For several of the load balancing algorithms, this work may be based on Load Estimates maintained for each of the servers. The Sender forwards requests to servers and updates Load Estimates and Session State as needed. The Receiver also receives responses sent by servers. The client to receive the response is identified by the Session Recognition module which obtains this information by querying the Session State. The Sender then sends the response to the client and updates Load Estimates and
Session State as needed. The Trigger module updates Session State and Load Estimates after a session has expired.

**SYSTEM REQUIREMENTS:**

**HARDWARE REQUIREMENTS:**

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

**SOFTWARE REQUIREMENTS:**

- Operating system : Windows XP.
- Coding Language : JAVA