An Efficient Server Load Balancing using Session Management

S.Tharani¹, Balika.J.Chelliah², Dr.J.Jagadeesan³

¹M.Tech, Computer Science and Engg, Ramapuram campus, SRM University, Chennai-600089.
²Asst.prof, Ramapuram campus, SRM University, Chennai-600089.
³Prof, Ramapuram campus, SRM University, Chennai-600089.

ABSTRACT

Load balancer allocates the work to the clusters of SIP server. The several load balancing algorithms for distributing Session Initiation Protocol (SIP) request to a cluster of SIP servers. The load balancer algorithm Transaction Least Work Left is used to allocate work to least values of the servers. It is combine knowledge of the SIP. Two types of state exist in SIP. The first, session state, is created by the INVITE transaction and is destroyed by the BYE transaction. The session-oriented nature of SIP has important implications for load balancing. The SIP is a protocol of growing importance, with uses for VOIP, IPTV, audio conferencing, instant messaging. 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.

Keywords—Load balancing, Server, Session Initiation Protocol (SIP)

1. INTRODUCTION
1.1 SIP PROTOCOL:
SIP is a signaling (control-plane) protocol designed to establish, modify, and terminate media sessions between two or more parties. The core IETF SIP specification is given in RFC 3261, although there are many additional RFCs that enhance and refine the protocol. 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 such as RSVP that is considered outside the scope of the protocol [5]. As another example, SIP can run over many protocols such as UDP, TCP, TLS, SCTP, IPv4, and IPv6.

1.2 SIP SERVER:
SIP Servers are essential network elements that enable SIP endpoints to exchange messages, register user location, and seamlessly move between networks. SIP Servers enable network operators to install routing and security policies, authenticate users and manage user locations.
The SIP baseline specification RFC3261 (previously RFC2543bis) divides SIP [5]

Server functionality into the following parts:
SIP Registrar Server—handles location registration messages.
SIP Redirect Server—returns “contact this address” responses.
SIP Proxy Server—forwards SIP requests and responses.

1.2.1 REGISTRAR SERVER
The SIP standard defines a registrar server as “a server that accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles”. REGISTER requests are generated by clients in order to establish or remove a mapping between their externally known SIP address(es) and the address(es) they wish to be contacted at. The REGISTER request can also be used to retrieve all the existing mappings saved for a specific address.

1.2.2 REDIRECT SERVER
Redirect server functionality is the simplest of the three functionalities. A redirect server receives SIP requests and responds with 3xx (redirection) responses, directing the client to contact an alternate set of SIP addresses. The alternate addresses are returned as Contact headers in the response message.

1.2.3 PROXY SERVER
The SIP standard defines SIP proxies as “elements that route SIP requests to User Agent Servers (UAS) and SIP
responses to User Agent Clients (UAC). A request may traverse several proxies on its way to a UAS. Each will make routing decisions, modifying the request before forwarding it to the next element. Responses will route through the same set of proxies traversed by the request in the reverse order."

A proxy server is designed to be mostly transparent to UAs. Proxy servers are allowed to change messages only in specific and limited ways. For example, a proxy is not allowed to modify the SDP body of an INVITE. Apart from a few exceptions, proxies cannot generate requests at their own initiative. Therefore a proxy cannot terminate an existing call by generating a BYE request.

The SIP specification defines two types of SIP proxies:
- Stateful proxy
- Stateless proxy

2. RELATED WORK

2.1 MESSAGE PROCESSING:
User agents are further decomposed into User Agent Clients (UAC) and User Agent Servers (UAS), depending on whether they act as a client in 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 (e.g., INVITE, BYE, CANCEL, etc.) and at least one response to that request. Responses may be provisional, namely, that they provide some short-term feedback to the user (e.g., 100 TRYING, 180 RINGING) to indicate progress, or they can be final (e.g., 200 OK, 407 UNAUTHORIZED). The transactions only completed when a final response is received, not provisional response. A SIP session is a relationship in SIP between two user agents that lasts for some time period; in VoIP, a session corresponds to a phone call. This is called a dialog in SIP and results in state being maintained on the server for the duration of the session. For example, an INVITE message not only creates a transaction (the sequence of messages for completing the INVITE), but also a session if the transaction completes successfully. A BYE message creates a new transaction and, when the transaction completes, ends the session. Fig. 2 illustrates a typical SIP message flow, where SIP messages are routed through the proxy. Nodes, the distributions of occupancy across the cluster are balanced, resulting in greatly improved response times. The naive approaches, in contrast, lead to imbalances in load. These imbalances result in the distributions of occupancy that exhibit large tails, which contribute significantly to response time as seen by that request. In this example, a call is initiated with the INVITE message and accepted with the 200 OK messages. Media is exchanged, and then the call is terminated using the BYE message [1].

2.2 LOAD BALANCING ALGORITHMS:
This paper introduces and evaluates several novel algorithms for balancing load across SIP servers. In addition, the best-performing algorithm takes into account the variability of call lengths, distinguishing transactions [2].

2.2.1 Call-Join-Shortest-Queue (CJSQ) tracks the number of calls (in this paper, we use the terms call and session interchangeably) allocated to each back-end server and routes new SIP calls to the node with the least number of active calls.

2.2.2 Transaction-Join-Shortest-Queue (TJSQ) routes a new call to the server that has the fewest active transactions, rather than the fewest calls. This algorithm improves on CJSQ by recognizing that calls in SIP are composed of the two transactions, INVITE and BYE, and that by tracking their completion separately, finer-grained estimates of server load...
can be maintained. This leads to better load balancing, particularly since calls have variable length and thus do not have a unit cost.

2.2.3) Transaction-Least-Work-Left (TLWL) New load balancing algorithm is based on assigning calls to the servers, which has (estimated) least amount of work assigned but not yet completed (Transaction Least-Work-Left). TLWL takes advantage of the observation that INVITE transactions are more expensive than BYE transactions. On our platform, a 1.75:1 cost ratio between INVITE and BYE results in the best performance. We implement these algorithms in the implementation for SIP server configured as a load balancer. 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. [2]

3. IMPLEMENTATION OF LOAD BALANCING

Fig. 2 depicts our overall system. User Agent Clients send SIP requests (e.g. INVITE, BYE) to our load balancer, which then selects a SIP server to handle each request. The Transaction-Least-Work-Left load-balancing algorithms presented in this paper are used to choose which SIP server to handle a request. Servers send SIP responses (e.g. 180 TRYING) to the load balancer, which then forwards the response to the client.

4. MODULES IN THE SYSTEM

➢ Neighbors Node Discover
➢ Load Balancer Design and Working
➢ Client - Server Communication using Load Balancer

4.1 NEIGHBORS NODE DISCOVER

The Neighbors Node Discover discovery is used to find the number of clients available in the network. When a new client is added into the network it will be updated to the neighbor list of all the available clients to identify them and communicate to the other entity.

The figure 3 explains that when the client CC444 enters into the network that particular information will be added to all other clients (CC111, CC222 and CC333) present at that particular time.
4.2 LOAD BALANCER DESIGN AND WORKING:

This section describes our implementation. Fig. 4 illustrates the structure of the load balancer. The rectangles represent key functional modules of the load balancer, while the irregular shaped boxes represent state information that is maintained. The arrows represent communication flows. The receiver receives requests that 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, which is implemented as a hash table. The Trigger module updates Session State and Load Estimates after the session has expired.

4.3 CLIENT - SERVER COMMUNICATION USING LOAD BALANCER

When the client1 needs to communicate to other client first the INVITE request from the client is sent to the load balancer. Server List and status (Workload) of the server has been updated in the Load balancer for each server. Load Balancer forwards the Client Request to the server which one is having the least workload. Server Status (workload and server disconnection) to be updated to the load balancer again.

5. PSUDOCODE FOR THE IMPLEMENTATION

Loadbalancer(Req from client, call_id)

If(Req==INVITE)
Server S=Execute TLWL Algorithm to select the server
Forward the request to the server S
Establish the session between client and server
Send 200 response to client
Put entry in the Active table
Increment the load count of server S
Update the load in to the table
Else if(Req==BYE)
Check if the session is active
S=Get the server for the session for the call_id
If(session==Active)
Terminate the session
Move the client entry to expired table
Decrement the load count of the server S
Update the load in to the table
Else if(Req= chat or voice chat or upload)
S=Get the server for the session for the call_id
If(session==Active)
Forward the req to server S
Else
Throught the exception that INVITE is not provided early
Else
Throught the exception that INVITE is not provided early
END

PSUDOCODE FOR LOAD BALANCER

Server(Req from load balancer,call_id)
If(Req==chat or voicechat or upload)
Get the desination address from the req
Search for destination next hop
Forward the req to the destination next hop
Else
Send error response
END

PSUDOCODE FOR SERVER

6. ADVANTAGES

- Response Time has been reduced
- Increase of Throughput
- Heterogeneous Back Ends
- Load Balancer Capacity

7. SUMMARY AND CONCLUSION

This paper introduces three novel approaches to load balancing in SIP server clusters (CJSQ, TJSQ, and TLWL.). We present the design, implementation, and evaluation of a load balancer for cluster based SIP servers. Our load balancer performs session-aware request assignment to ensure that SIP transactions are routed to the proper back end node that contains the appropriate session state. By combining knowledge of the SIP protocol, recognizing variability in call lengths, distinguishing transactions from calls, and accounting for the difference in processing costs for different SIP transaction types, load balancing for SIP servers can be significantly improved.

REFERENCES