Technical Disclosure Commons

Defensive Publications Series

June 2020

INTENT BASED LOAD-BALANCING FOR VOICE OVER INTERNET PROTOCOL (VOIP) ELEMENTS

Rajarshee Dhar

Rajesh Kalagarla

INTENT BASED LOAD-BALANCING FOR VOICE OVER INTERNET PROTOCOL (VOIP) ELEMENTS

AUTHORS:

Rajarshee Dhar
Rajesh Kalagarla

ABSTRACT

Presented herein is an intelligent call distribution/load balancing solution that performs distribution based on the type of call. The solution determines the type of call based on various factors and uses reinforced learning algorithms to select the element best suited for that type of call based on call-success-ratio for a particular call type (e.g., audio/video/fax/ etc.). This eliminates call failures, call delays and improves customer satisfaction. This solution can be extended to various other details of the call like dual tone multiple frequency (DTMF), codec, payload type, etc.

DETAILED DESCRIPTION

Currently, Session Initiation Protocol (SIP) proxies / load-balancers perform load balancing based on defined parameters such as weight, priority etc. Generally, elements which are part of the load balancing group may have different capabilities and maybe suited for a specific type of calls. In addition, there is a possibility that a particular element in the group may be vulnerable to a specific call flow that can result in all such calls failing when they are load-balanced to that particular element. Retry of those failed calls adds additional load on the overall system and adds more delay in call routing, negatively impacting customer satisfaction.

Moreover, current implementations of SIP proxies / load-balancers do not account for the nature of the element and its capabilities for a particular type of call. As a result, the best element is often not utilized for the call, which further results in unwanted outcomes like call failures, and overutilization of the elements.

With the introduction of multiple heterogeneous elements in today's Voice over Internet Protocol (VoIP) systems and high customer satisfaction requirements, simple call

distribution algorithms based on priorities and weights are not sufficient. There is a clear need for intent-based call distribution where the load-balancer is expected to intelligently distribute calls to the elements which are best suited for that type of call.

The Solution

Presented herein is a solution that simplifies the current load balancing schemes. This solution takes into account the type of the call and its specific success rate with a specific server element when to selecting the suitable element for that type call.

For instance, Alice calls Bob in the below call flow:

Alice--> Call-agent - --------- LB (our device of interest) ------- (Srv1, Srv2, Srv3…..… Srvn)- -----SBC- ---Call-Agent ----Bob

This solution is focused on the load balancing (LB) function/entity, which has n number of downstream elements of different capabilities. LB needs to select a server from the set (Srv1, Srv2….Srvn).

Contrary to the typical load-balancing scheme, this solution employs mechanisms to identify the call-type/nature of call that Alice is trying to make based on known methodologies for call-type identification (Session Description Protocol (SDP), historical call details of the same calling and called number, querying an internal enterprise database or external telco database) and finally choosing the element among the set (Srv1, Srv2…Srvn) best suited for the call.

In some of the calls, the load-balancer will choose Srv1, in some Srv2, and in some Srvn, essentially balancing the load, based on call-type identification.

Similarly, Srv1 can further select among its own downstream elements (S1, S2, S3…Sn) and those elements can also further choose their best downstream element based on the desired call-type essentially creating a chain of servers selecting their best downstream elements resulting in the selection of the most efficient path suited for the call.

This decision making is done even before actual media is negotiated. Based on an incoming SIP INVITE for a call, the details required to identify the call type is fetched and

input to a model, which in turn predicts the type of the call. An example of this is described further below.

The load-balancer will selects between a set of downstream elements (S1, S2, S3….Sn) of varying capabilities. Once the call-type is determined, as explained above through reinforced learning, the best element is selected to send the call. Based on a data set and call success ratio associated with that specific call-type, the load-balancer will decide and choose one of the elements to which to send the call.

It may look like this:

INVITE with calling and called number, SDP ---> [Call-type Identification]-> Pass result-> [Downstream element selection] -> INVITE goes out

where [Call-type Identification]-> Pass result-> [Downstream element selection] will happen inside the load-balancer using machine learning.

With powerful servers/SBCs/Proxies moving away from specific hardware boxes to powerful shared hardware utilization platforms (birtual machines), running machine learning algorithms on such devices is no limitation. In fact, with larger data sets, the required configuration/human intervention will be minimal (to none) on these devices in the near future.

As a further example, there may be a device A that is a video-enabled device and has a very high success rates for video calls. Similarly, there may be another device (device B) which supports audio calls and has a high success rate for a specific audio codec. Likewise, there is a device C which is a fax capable device and has high success rate of fax calls. In such a scenario, the load-balancer will send fax calls to device C, audio calls to device B and video calls to device A.

Moreover, the load-balancer will feed to the model types of calls and corresponding success rates. This model-based data will help the load-balancer to self-learn the overall behavior of the solution. The model will grow over the time and the logical decisions taken during course of time will become more accurate, thus avoiding manual intervention of administrators to tweak the call routing logic to reduce the failures.

The device in presented herein is not a call-agent but can be a proxy or a back-to-back user agent (B2BUA). Call-agents may store the capabilities of its devices during registration and use them for routing. The device presented herein does not have any concept of registration so it does not necessarily poll its downstream elements for the same. Furthermore, downstream elements may be heterogeneous devices from different vendors and may not share the information with the device.

Thus, the load-balancer will decide the capability in-dialog, in addition to analytics done on historical call records from the specific calling and called-number set, and decide which is the right candidate for the call based on historical call-success-ratio.

There are reinforced learning algorithms involved to correctly find a pattern among the data set and formulate a model that can predict the call-type. A call-agent already is overburdened with maintaining in-memory registration and routing information, in addition to various call interworking handling (CPU and memory intensive) and is. Thus, the right fit is a load0balancer like a proxy or a B2BUA.

Below is an example that can be created for each server element and explaining how updates are made with the call final result.

{
Element 1: Call-type : Audio: Codec: G.722: CSR: 80%
Codec: G.711: CSR: 82%
Codec: OPUS: CSR 40%

Fax: Codec: T.38: CSR: 96 %
Codec: G711 Passthrough: 70%

Video: Codec: H264: CSR: 90%

Element 2: Call-type : Audio: Codec: G.722: CSR: 40%
Codec: G.711: CSR: 92%
Codec: OPUS: CSR 70%

Fax: Codec: T.38: CSR: 56 %
Codec: G711 Passthrough: 90%

Video: H264: CSR: 45%

}

Solution Design

The solution may be designed with 3 stages of operation:

1. Identifying the incoming call-type on the server.
2. Choosing the best downstream element for specific call-type.
3. Decision making with respect to current statistics.

1. Identification of call-type for effective element selection for call routing

The call-type determination is done considering the below factors in order.

a) **SDP of the SIP message:**

Looking at the incoming SDP message's m-lines, the type of incoming call can be determined, as well as its respective capabilities.

For example:

INVITE sip:9000@10.64.86.113:5060 SIP/2.0
Via: SIP/2.0/UDP 10.64.86.232:8872;branch=z9hG4bK-5005-1-0
From: "Rajadhar" <sip:8000@10.64.86.232:8872>;tag=1
To: "SIPp" <sip:9000@10.64.86.113:5060>
Call-ID: 1-5005@10.64.86.232
CSeq: 1 INVITE
Contact: <sip:8000@10.64.86.232:8872;transport=UDP>
Allow:
ACK,BYE,CANCEL,INFO,INVITE,OPTIONS,PRACK,REFER,NOTIFY,UPDATE
Accept: application/media_control+xml,application/sdp,multipart/mixed
Max-Forwards: 69
Content-Type: application/sdp
Content-Disposition: session;handling=required
Content-Length:281

v=0
o= Cisco Systems 91257780 1 IN IP4 10.64.86.232
s=-
c=IN IP4 10.64.86.232
t=0 0
m=audio 6001 RTP/AVP 18 101. ---> To capture media type
a=rtpmap:18 G729/8000

a=fmtp:18 annexb=no
a=rtpmap:101 telephone-event/8000 ---> To capture dtmf mechanism
a=fmtp:101 0-15
a=sendrecv
a=maxptime:20

Clearly in this SIP message, we know that the incoming call-type is audio because of "m=audio 6001 RTP/AVP 18 0 8 101". If the call were a video call, we would have received "m=video XXX". However, conclusion that this call is an audio call directly here may not be correct. It may happen that the call started with audio but later escalated to video. Consider that a customer joins an online conference meeting with an audio bridge and later decides to share video during the conference. More details about the type of the call are needed, so we move to the next step.

From this module, some information is extracted that may assist in later decisions.

Example:

Codec: If we mark the call-type as an audio call, then we can use the codec information to find the best element suited for the audio call for that codec (G729 in this case).

Calling number and Called number: We will use this information to further verify the call type in the next layer.

Note: There may be a scenario that the incoming INVITE message does not have an SDP field, so we will skip this layer and move on the next layer (call-type associated with a number) to identify the call-type.

b) Call-type associated with a number:

Based on historical data collected over time, the system knows call-type associations with telephone numbers. Since the server participates in the call, the server

knows the call-types of successful calls. The server keeps a track of the call until the call is disconnected. Anytime call information is changed/updated (audio to video as discussed above), the server is going to update the "Number-Call Type" database based on historical call statistics.

Example:

Historical Call statistics:

Calling Number	Called Number	Audio	Video	Application
+91 915978 5000	+49 8978 673451	120	0	0
+91 45677 89900	+1 415 645 8770	6	30	0
+91 45677 89900	+44 67890 45623	40	2	70
+91 78871 67545	+ 91 56345 76768	1007	56	3
+91 78871 66772	+91 78871 67545	50	245	8

From the above data, the server can use the counters (audio, video and application) as data points to decide the call-type and update the “Number-Call Type” DB as follows:

// Sample update to the table

update set number_call_type call_type = 'audio' where calling_number = 'xxxxx' and called_number = 'yyyy' \G;

The Number-Call Type Database may take the form of:

Calling Number	Called Number	Call-type
+91 915978 5000	+49 8978 673451	Audio
+91 45677 89900	+1 415 645 8770	Video
+91 45677 89900	+44 67890 45623	Application
+91 78871 67545	+ 91 56345 76768	Audio
+91 78871 66772	+91 78871 67545	Video

With more data points in the historical call statistics, the accuracy of the model to set the call-type will improve.

The system can later query the database at stage 2 to determine the call-type for a specific calling-number and called-number as follows:

// Sample query to retrieve call-type

select number_call_type from table_name where calling_number = 'xxxxx' and called_number = 'yyyy' \G;

c) Number type association from Telco / Cloud application:

This step involves learning the nature of the number and building a local database. This is the final resort to determine the call-type and for use successful call routing in the first attempt. This may occur when either steps a) or b) above are not able to determine the call-type, such as when routing to a particular number for the first time and the SDP is not present on the INVITE message.

The system will query the Service Provider database to identify the capabilities for that particular number through Hypertext Transfer Protocol/Hypertext Transfer Protocol Secure (HTTP/HTTPS). Based on the details, the system can update the Number-Call Type database as described above.

If the number is not found in any of the above identification pattern, calls will be routed based on provisioning done by the administrator.

2. Selecting the best downstream element based on call-type

Once we have identified the call-type, the best downstream element is identified for sending the call. For this, an element historical database is consulted. This database is updated after every call completion.

Example of Element historical Database:

Element IP address	Call-type	Codec	CSR in % (Call-Success Rate)	Memory utilization in %	CPU utilization in %
72.56.187.82	Audio	G729	90	24	20
72.56.182.81	Audio	G711u	99	54	40
72.56.187.82	Video	H.264	20	24	20
72.56.182.191	Video	H.264	97	30	20
72.56.187.82	Audio	G711u	89	24	20
72.56.182.191	Audio	G729	10	30	20
72.56.182.191	Audio	G711u	23	30	30

From the above table the server can create metadata, such as:

{
72.56.187.82: Call-type : Audio: Codec: G.729: CSR: 90%
Codec: G.711: CSR: 89%

Video: Codec: H264: CSR: 20%

72.56.187.191: Call-type : Audio: Codec: G.729: CSR: 10%
Codec: G.711: CSR: 23%

Video: Codec: H264: CSR: 97%

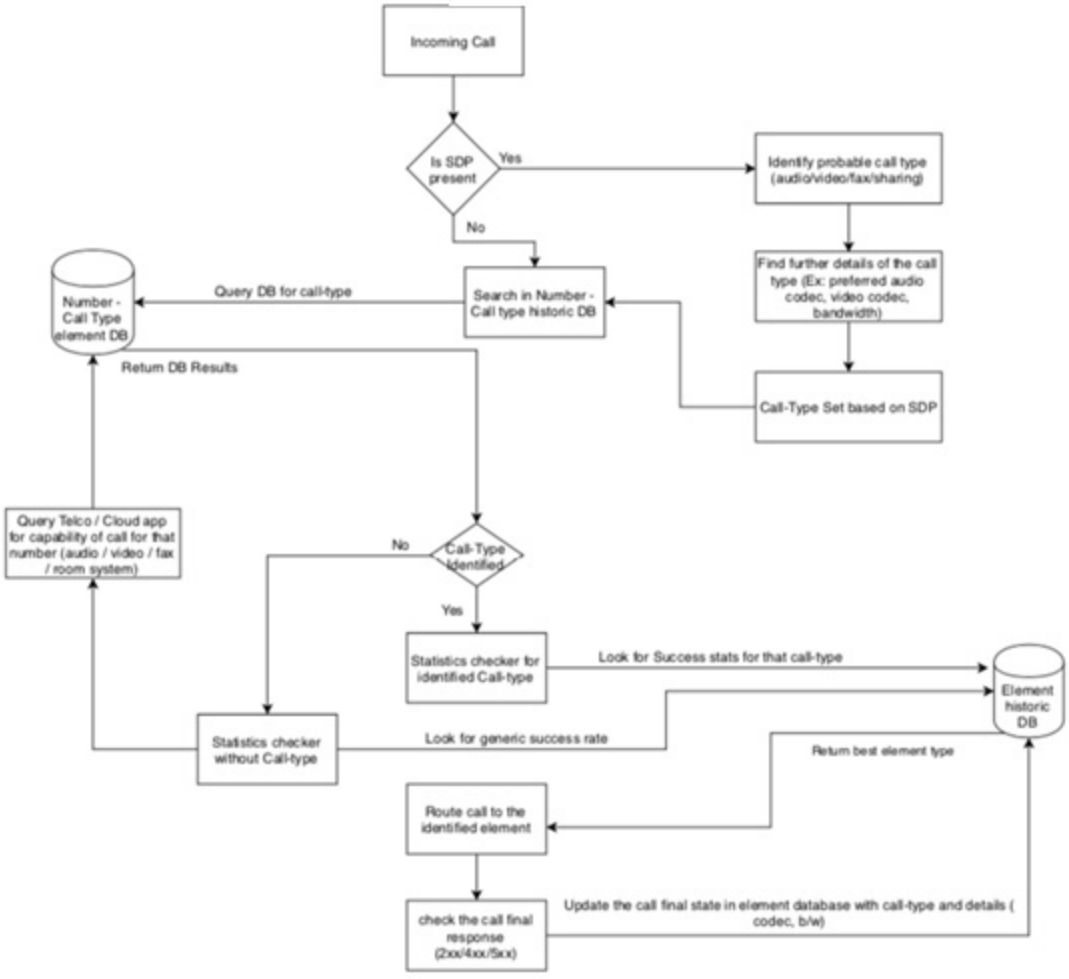
}

So clearly, if the call is an audio call with codec G729, the system will automatically choose an element with IP 72.56.187.82 address as it has the highest CSR for that type of call. Similarly, for a video call, the system will automatically choose element with IP address 72.56.187.191 for the call. With more calls, the precision of selecting the right element increases and thus will enable the system to always select the best element suited for that specific type of call.

3. Decision making with respect to current stats

Once the best element is identified, the system needs to make sure that element is capable of taking on the call. The system may check the CPU utilization and memory utilization of the downstream element by periodic polling it through SIP OPTIONS or some other similar message. If the utilization of the resources of the downstream element is well below a threshold, then call will be sent to that element. If the utilization of the resources exceeds the threshold, then the server needs to select the second best element for that type of the call by querying the database.

Set forth below is an example flow chart for this solution.



Example Use-Cases

Some of the use cases for this call-type identification as follows.

1. Consider a user who dials an audio call, talks for some time, and later switches into a video call. The moment he switches to a video call, he starts facing issues or worse, his call drops. If the call is somehow active, but has video issues, in that case, the user can either notify the intelligent load-balancer through a DTMF/unsolicited message to switch to a suitable video path or the caller disconnects and tries again. The next time the caller starts with audio+video call. Now, the load-balancer will select the best video capable path as it identified the call as a video call.
2. Identifying call-type is an important task. Looking into incoming SDP information is one way and looking at the historical call details between the caller and callee is another way. Furthermore, the originator (caller) and call receiver (callee) capabilities can be queried directly from an internal database (within the enterprise) or from the Telco database (external). For instance, a video conference call to a destination (callee) from a video conference endpoint (caller) indicates that the incoming call will be bandwidth-intensive. Thus, the load-balancer will select the downstream element that guarantees Quality of Service (QoS) and high bandwidth for the call. The load-balancer identifies the call-type as a video conference call and already knows from its dataset the next element to which it should route the call, which is ideal for the video conference call and can guarantee QoS.

3. The load-balancer identifies the pattern of a user dialling and leverages machine learning algorithms to learn the behaviour of the calls. Over time, the successful prediction of call type increases. There is always a clear dialling pattern between the users in an enterprise. For instance, Alice always sends a fax to Bob. Therefore, if a call originated from Alice's number to Bob's number, that means the call is a fax call. The load-balancer using multiple mechanisms, as presented herein, to identify the call-type and selects the element which has demonstrated a high success rate for the fax calls made from Alice to Bob in the past. If such data is not present, the load-balancer will select the element that has a high success rate for fax calls.

4. Consider a scenario where the immediate downstream elements of the load-balancer are SIP proxies, call controllers, SIP-TDM gateways, etc. Cathy decides to dial to Dennis using her new phone that supports an OPUS codec. When the call comes to the load-balancer, it can select any element from its pool using some static load-balancing schemes. The problem is that selecting any of those listed downstream elements is not efficient because it may be desirable to not consume resources of a particular element for an audio-only call. Furthermore, if the call requires advanced audio interworking capabilities (supplementary services, midcall signaling, various form of interworking), some elements may not the best element. For example, one element may not support the OPUS codec, so sending a call to that element will directly fail. Sending a call to SIP-TDM gateway may not be preferred because of OPUS transcoding to pulse code modulation (PCM) samples. Only one particular element may be the best choice. Using traditional mechanism, the call had to fail 3 times (at least), and then it would select the particular element, which will add unnecessary delay to the caller. This problem can be easily solved by this solution because the model already knows about the call-type and based on the incoming call-type, it selects the element best suited for an audio call with OPUS codec with the highest Call Success Ratio, on the first attempt.

Most of the issues faced today are not because of the limitation of endpoints but because of the internal devices involved and the path traversed. This solution mitigates this problem by selecting the most optimal device suited for the specific call-type based on reinforced learning algorithms. These algorithms operate on call behaviour based on historical statistics, call-type associations, and device capability sets.

In summary, presented herein is an intelligent call distribution/load balancing solution that performs distribution based on the type of call. The solution determines the type of call based on various factors and selects the element best suited for that type of call based on call-success-ratio for a particular call type (audio/video/fax/ etc.). This eliminates call failures, call delays and improves customer satisfaction. This solution can be extended to various other details of the call like dual tone multiple frequency (DTMF), codec, payload type etc. The efficiency of the system increases manifold as it is fed with more data points.