Internet Engineering Task Force IPTEL WG
Internet Draft
J.Rosenberg,H.Salama
dynamicsoft, Cisco
draft-rs-trip-gw-01.txt
July 14, 2000
Expires: January 2001

Usage of TRIP in Gateways for Exporting Phone Routes

Abstract

This document describes a new application of the Telephony Routing over IP (TRIP) protocol. TRIP was engineered as a tool for inter-domain exchange of telephone routing information. However, it can also be used as a means for gateways and soft switches to export their routing information to a Location Server (LS), which may be co-resident with a proxy or gatekeeper. This LS can then manage those gateway resources. We discuss the motivations for this application, the reasons why other mechanims, such as the SIP REGISTER method, are not appropriate, and then show how a minimal subset of TRIP is needed for this application.

1 Introduction

In typical VoIP deployments, a service provider runs a network composed of numerous gateways, softswitches, and proxy servers. Generally, gateways (both composed and decomposed) are distributed geograpically throughout the network. When a gateway terminates a call to a number, cost to the service provider is incurred. This cost is partly dependent on the cost of making a call over the PSTN from the gateway to the destination number. By placing gateways in numerous locations over the globe, the service provider can be sure it can terminate calls to the PSTN with minimal cost.

When calls arrive at the network (either from customers of the provider, or from peer providers desiring termination service), they are first sent to a proxy which acts as a routing engine. Based on the knowledge of available gateways, their capacity (in terms of circuit and DSP resources) and other attributes, and the telephone numbers each gateway can terminate calls to, the proxy forwards the call to the appropriate gateway. In essence, the LS/proxy is responsible for managing the real-time resources made available by a set of gateways.

This configuration is depicted graphically in Figure 1.

There are several problems that must be solved in order to enable this scenario:

- o Often, the routing tables in the routing proxies are manually entered. This makes configuration more complex, particularly for large networks with hundreds or even thousands of gateways. In the ideal scenario, each gateway is configured with the numbers it can terminate calls to, and with the address of the routing proxies. The gateways then push their routing information to the proxy. No standard mechanism has been specified to do this.

- o It is important for the routing proxy to be aware of failures of gateways. This way, an alternate gateway can be selected for new incoming calls. This requires some kind of keepalive between the gateways and the routing proxy. No standard mechanism yet exists for this keepalive.

- o The routing proxy will need to route call setup requests to the gateways based on dynamic attributes of those gateways. In particular, the available capacity, measured in terms of both circuit resources and coding resources, is used to properly route calls. The proxy can, for example, perform load balancing by forwarding call setups to the most lightly loaded gateway among a set. No standard mechanism has been specified to do this.

```
                                                +---------+
                                                |         |
                                                |   GW    |
                                             >  +---------+
                                           //
                                         //
                                       //       +---------+
                                     //         |         |
                                   //           |   GW    |
                                 //             +---------+
                               //
              +----------+   //                                TO PSTN
              |          |  /                   +---------+
              | Routing  |          ------->    |         |  ----->
     -------->| Proxy    |   -------            |   GW    |
              |          |  --                  +---------+
              |          |    --
              +----------+      --
                                  ---           +---------+
                                     --         |         |
                                       --       |   GW    |
                                         --     +---------+
                                           -->

                                                +---------+
                                                |         |
                                                |   GW    |
                                                +---------+
```

Figure 1: Gateway and LS Configuration

This document discusses how TRIP [1] can be used to solve these two problems. The first section reviews other mechanisms for accomplishing this - namely the SIP [2] REGISTER method, and discusses why TRIP is a much better solution. We assume the reader is familiar with TRIP. An overview of its architecture can be found in [3].

2 Why not SIP REGISTER?

The SIP REGISTER method has frequently been proposed as a solution for these two problems. This is due, in part, to the similarity of the REGISTER method to the H.323 [4] RAS messages. In H.323, RAS messages are used to allow gateways to register telephone number

prefixes with a gatekeeper. Many assume that SIP REGISTER should therefore play a similar role.

Certainly, the REGISTER mechanism is close to this functionality. REGISTER allows clients to send address bindings (including for telephone numbers) to a proxy. Registrations are also periodically refreshed, allowing a proxy to determine if the address binding becomes stale, perhaps due to a crash or device failure. The refresh timer (present in the Expires header) can even be negotiated by the proxy.

However, the SIP REGISTER mechanism is different from the desired mechanisms for gateways in many respects:

- The REGISTER mechanism is used to bind a single incoming URI to one or more outgoing URIs. In the case of a telephony gateway, the binding is between a set of telephone prefixes to the address of a gateway. This is a significantly different binding, and cannot be represented with the syntax or semantics of a SIP REGISTER request.

- The keepalive mechanism in REGISTER refreshes the *binding*, not the status of the UA performing the registration. The bindings must be sent each time to keep them alive. In the case of a gateway, the keepalive is for the state of the gateway, not for the routes the gateway terminates. The semantics of REGISTER would need to be completely changed in order to support this different semantic.

- There are properties associated with gateway routes that are not associated with URIs. For example, a route may have information like capacity (how many simultaneous calls can be terminated), which does not make much sense for a property of a URI.

- Because gateways can handle so many calls, it is important for liveness information to be conveyed very frequently, on the order of seconds. SIP registrations are not meant to be sent that frequently; they can be fairly large messages (particularly if they need to contain the routes when refreshed), resulting in uneeded overheads.

For these reasons, we do not believe the SIP REGISTER request is the right tool for gateway route propagation and gateway keepalives.

3 Why TRIP?

TRIP was engineered as a tool for interdomain route exchange. It is

not a simple protocol, and at first glance, does not seem appropriate for application in a gateway.

However, TRIP provides exactly the features needed to solve the problem at hand. TRIP allows one entity (in this case, a gateway) to convey to another (in this case, a proxy) a set of telephone routes which terminate through it. These routes are represented by telephone number prefixes. In TRIP, the routing tables are exchanged once. Only when they change are updates sent. This is exactly the capability needed for a gateway to send its routing information to a proxy.

TRIP also supports a keepalive between peers. This keepalive is a short message, sent fairly frequently. It does not contain routing information. The period of the keepalive is negotiated once at startup time. If one of the entities crashes, the other flushes all routes received from it. This, too, is exactly the mechanism needed for keepalives in a gateway.

TRIP can contain attributes describing a route. New attributes can easily be added. The available capacity of a route is needed by the proxies to properly load balance and route to a set of gateways. This capacity can be expressed as an attribute.

Another advantage of using TRIP here is that it makes the redistribution of local gateway reachability information into inter-domain TRIP straightforward, because it's the same protocol.

While it is true that TRIP is complex, almost all of this complexity is related to the processing of routes *received* from other peers. An element, such as a gateway, which only *sends* routes to a peer (the proxy), does not need to implement any of those functions. In particular, any processing related to aggregation, TRIB updates, route propagation and advertisement, handling of transitivity and unknown routes, or the decision process need not be implemented. The resulting set of functions are very small. They are composed of only:

- o The initial OPEN phase, exchange of keepalive timers, and the process of bringing up the state machine.

- o Sending of an UPDATE containing the routes and parameters of the gateways.

- o Sending of a periodic keepalive.

Its worth noting that these functions are not substantially more complex than sending a periodic REGISTER.

The rest of this document is organized as follows. Section 4

discusses a new attribute, circuit capacity, and section 5 discusses another new attribute, DSP capacity. These new attributes contain dynamic capacity of gateways that can be propagated to the LS managing those gateways. Section 6 describes the processing rules needed for a gateway, and section 7 discusses some of the processing needed in an LS.

4 CircuitCapacity Attribute

Mandatory: False.
Required Flags: optional, non-transitive
Potential Flags: None.
TRIP Type Code: TBD.

The circuit capacity attribute is used only between a gateway and its peer LS responsible for managing that gateway. It is for this reason that this attribute is non-transitive. If it is received in a route, it SHOULD NOT be propagated unless the LS is sure that it is relatively static.

The circuit capacity identifies the number of PSTN circuits that are currently available on a route to terminate calls. The number of additional calls sent to that gateway for that route can not exceed the circuit capacity. If it does, the signaling protocol will likely generate errors, and calls will be rejected.

Circuit capacity is measured in integral number of calls. It changes very dynamically.

4.1 CircuitCapacity Syntax

The CircuitCapacity attribute is a 4-octet unsigned numbeic value. It represents the number of circuits remaining for terminating calls to this route. This attribute represents a potentially achievable lower bound on the number of calls which can additionally forwarded on this route.

4.2 Route Origination and CircuitCapacity

Routes MAY be originated containing the CircuitCapacity attribute. Since this attribute is highly dynamic, changing with every call, updates MAY be sent as it changes. However, it is RECOMMENDED that a gateway originating routes with this attribute use thresholds, and that routes are re-originated only when the attribute moves above or

below a treshold. It is also RECOMMENDED that the thresholds in each direction (going above a threshold and going below a threshold) be different, thus achieving a form of hysterisis. Both of these measures help reduce the messaging load from route origination.

4.3 Route Selection and CircuitCapacity

The CircuitCapacity attribute MAY be used for route selection. Since one of its primary applications is load balancing, an LS will wish to choose a potentially different route (amonst a set of routes for the same prefix) on a call by call basis. This can be modeled as re-running the decision process on the arrival of each call. The aggregation and dissemination rules for routes with this attribute ensure that re-running this selection process never results in propagation of a new route to other peers.

4.4 Aggregation and CircuitCapacity

An LS MUST aggregate routes to the same prefix which contain a CircuitCapacity attribute. This guarantees that if the decision process is rerun, the route that is disseminated to peers is unchanged.

4.5 Route Dissemination and CircuitCapacity

Routes SHOULD NOT be disseminated with the CircuitCapacity attribute. The attribute is meant to reflect capacity at the originating gateway only. Its highly dynamic nature makes it inappropriate to disseminate in most cases.

5 DSPCapacity attribute

Mandatory: False.
Required Flags: optional, non-transitive
Potential Flags: None.
TRIP Type Code: TBD.

The DSPCapacity attribute is used only between a gateway and its peer LS responsible for managing that gateway. It is for this reason that this attribute is non-transitive. If it is received in a route, it SHOULD NOT be propagated unless the LS is sure it is relatively static in value.

The DSPcapacity identifies the amount of DSP resources, in MIPS, that

are currently available on a route to terminate calls. The metric should be considered as only an approximate. The MIPS are computed based on a specific processor (TBD); other processors will need to perform a conversion to obtain this normalized parameter. It is assumed that the LS is aware of the DSP resource requirements for each call, based on the set of codecs indicated in the messages routed by the LS/proxy.

> There is lots of handwaving here. Can we usefully define this metric? How to determine how much DSP resources are really required to terminate a call? Also, the codec used may not be known at the time the message is to be routed. This can happen with both SIP (when the SDP in the INVITE is empty), and H.323. How to handle this?

DSP capacity is measured in integral number of MIPS. It changes very dynamically.

5.1 DSPCapacity Syntax

The DSPCapacity attribute is a 4-octet unsigned numbeic value. It represents the number of MIPS remaining for terminating calls to this route.

5.2 Route Origination and DSPCapacity

Routes MAY be originated containing the DSPCapacity attribute. Since this attribute is highly dynamic, changing with every call, updates MAY be sent as it changes. However, it is RECOMMENDED that a gateway originating routes with this attribute use thresholds, and that routes are re-originated only when the attribute moves above or below a treshold. It is also RECOMMENDED that the tresholds in each direction (going above a threshold and going below a threshold) be different, thus achieving a form of hysterisis. Both of these measures help reduce the messaging load from route origination.

5.3 Route Selection and DSPCapacity

The DSPCapacity attribute MAY be used for route selection. Since one of its primary applications is load balancing, an LS will wish to choose a potentially different route (amonst a set of routes for the same prefix) on a call by call basis. This can be modeled as re-running the decision process on the arrival of each call. The aggregation and dissemination rules for routes with this attribute ensure that re-running this selection process never results in propagation of a new route to other peers.

5.4 Aggregation and DSPCapacity

An LS MUST aggregate routes to the same prefix which contain a DSPCapacity attribute. This guarantees that if the decision process is rerun, the route that is disseminated to peers is unchanged.

5.5 Route Dissemination and DSPCapacity

Routes SHOULD NOT be disseminated with the DSPCapacity attribute. The attribute is meant to reflect capacity at the originating gateway only. Its highly dynamic nature makes it inappropriate to disseminate.

6 Gateway Operation

The protocol a gateway uses to advertise its E.164 reachability to the its domain's Location Server(s) (LS)is TRIP. The gateway operates in TRIP Send Only mode since it is only interested in advertising its reachability, but is not interested in learning about the reachability of other gateways and other domains. Also, the gateway will not create its own call routing database. Therefore, the gateway does not need a complete implementation of TRIP. A lightweight version of the protocol is sufficient. In this section we describe the operation of TRIP on a gateway. We refer to the protocol operating in this context as TRIP for Gateways, or TRIP-GW.

TRIP-GW is a stripped down version of TRIP, but still completely interoperable with normal TRIP speakers. It is an implementation profile, not an extension or incompatible reduction.

The reader is assumed to be familiar with TRIP. In our discussion we will skip most of the details common to both versions.

6.1 Session Establishment

When opening a peering session with a TRIP LS, an TRIP-GW gateway follows exactly the same procedures as any other TRIP speaker. The TRIP-GW gateway sends an OPEN message which includes a Send Receive Capability in the Optional Parameters. The Send Receive Capability is set by the gateway to Send Only. When the TRIP LS receives the gateway's OPEN message, it set its local policy such that no UPDATE messages are sent to the TRIP-GW gateway. The remainder of the peer session establishment is identical to TRIP.

6.2 UPDATE Messages

Once the peer session has been established, the gateway sends UPDATE messages to the TRIP LS with the gateway's entire E.164 reachability

and its ITAD topology.

If the gateway's E.164 reachability or its ITAD topology changes at any point in time, the gateway MUST generate UPDATE message(s) with the change. The frequency of successive UPDATE messages MUST follow the same rules specified for TRIP [1]. The TRIP-GW gateway MUST at least support all mandatory TRIP attributes.

If the gateway receives an UPDATE message from the TRIP LS, it MUST silently discard it as specified in [1]. No further action should be taken.

6.3 KEEPALIVE Messages

KEEPALIVE messages are periodically exchanged over the peering session between the TRIP-GW gateway and the TRIP LS as specified in Section 5.4 of [1].

6.4 Error Handling and NOTIFICATION Messages

The same procedures used with TRIP, are used with TRIP-GW for error handling and generating NOTIFICATION messages. The only difference is that an TRIP-GW gateway will never generate a NOTIFICATION message in response to an UPDATE message, irrespective of the contents of the UPDATE message. Any UPDATE message is silently discarded.

6.5 TRIP-GW Finite State Machine

When the TRIP-GW finite state machine is in the Established state and an UPDATE message is received, the UPDATE message is silently discarded and the TRIP-GW gateway remains in the Established state. Other than that the TRIP finite state machine and the TRIP-GW finite state machine are identical.

6.6 Call Routing Databases

A TRIP-GW gateway may maintain simultaneous sessions with more than one TRIP LSs. A TRIP-GW gateway maintains one call routing database per peer TRIP LS. These databases are equivalent to TRIP's Adj-TRIBs-Out, and hence we will call them Adj-TRIB-GWs-Out. An Adj-TRIB-GW-Out contains the gateway's E.164 reachability information advertised to its peer TRIP LS. How an Adj-TRIB-GW-Out database gets populated is outside the scope of this draft (possibly by manual configuration).

The TRIP-GW gateway does not have databases equivalent to TRIP's Adj-TRIBs-In and Loc-TRIB, because the TRIP-GW gateway does not learn routes from its peer TRIP LSs, and hence it does not run call route

selection.

6.7 Route Selection and Aggregation

TRIP's route selection and aggregation operations SHOULD NOT be implemented by TRIP-GW gateways.

7 LS Behavior

TRIP completely specifies the behavior of the LS as a TRIP speaker. However, the primary question is: is an LS connected to a gateway an internal or external peer of the gateway?

The most obvious choice, internal peer, is not the best choice. If an LS has ten peer GWs, all of them advertising reachability to 1408*, it will flood all ten routes to all other LSs in the same ITAD. This won't scale, because each LS in the ITAD will have to create a separate Adj-TRIB-In for each GW in that ITAD. In addition, it doesn't allow a SIP Proxy server or an H.323 GK to load balance among the GWs of its zone/subdomain.

A similar problem exists when an LS is an external peer to the gateways, and has direct peering relationships with one or more internal peers. However, an ingress LS to an ITAD does not perform aggregation. Only the egress aggregates routes.

Therefore, it is RECOMMENDED that the appropriate architecture is that the LS actually runs two instances of TRIP; one with an external peering relationship to the gateways, and the other with an internal peering relationship with one or more LS's within the ITAD. The interface between these instances is a local matter; routes are exported from one and imported to the other. This architecture is shown in Figure 2.

8 Conclusion

We have argued that the problem of managing a set of gateways from a location server is critical. This process of management includes propagation of routes, liveness determination, and propagation of available capacity for the purposes of load balancing. TRIP is ideally suited for these problems. As such, we propose here to define TRIP-GW, a subset of TRIP functionality (yet still 100% compatible with it) for use on gateways to perform this function.

9 Changes since -00

- o Added text to stress the value of this proposal for managing a

```
                                                              +-----+
                                                              |     |
  ...................................                       --| GW  |
  .                   +-------------.-----------+    ---   +-----+
  .                   | +--------+  .+--------+  |  --
  .                   | |TRIP    |  .|TRIP    |  +--       +-----+
  .                   |/|Instance|  .|Instance|--|         |     |
  .                   / |        |  .|        |--+-------- | GW  |
  .                  /| |        |  .|        |--|         +-----+
  .                 / | +--------+  .+--------+  +---
  .                /  |           LS.            |   ---   +-----+
  .               /   +-------------.-----------+      -- |     |
  .              /                  .                      | GW  |
  .  +----------+                   .                      +-----+
  .  |          |                   .
  .  |          |                   .
  .  |    LS    |                   .
  .  |          |                   .
  .  |          |                   .
  .  +----------+                   .                      +-----+
  .              \                  .                      |     |
  .               \                 .                   -- | GW  |
  .                \   +-------------.-----------+    --   +-----+
  .                 \  | +--------+  .+--------+  | ---
  .                  \ | |TRIP    |  .|TRIP    |  +-        +-----+
  .                   \| |Instance|  .|Instance|--|         |     |
  .                    \ |        |  .|        |--+---------| GW  |
  .                    | |        |  .|        |--|         +-----+
  .                    | +--------+  .+--------+  +---
  .                    |           LS.            |   ---   +-----+
  .                    +-------------.-----------+      -- |     |
  .       ITAD                       .                      | GW  |
  ...................................                       +-----+
```

Figure 2: LS Architecture for TRIP-GW

gateway cluster

- o Added attributes for circuit capacity and DSP capacity

- o Added section on LS operation, discussing aggregation issue

10 Authors Addresses

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
email: jdrosen@dynamicsoft.com

Hussein F. Salama
Cisco Systems
Mail Stop SJ-6/3
170 W. Tasman Drive
San Jose, CA 95134
email: hsalama@cisco.com

11 Bibliography

[1] J. Rosenberg, H. Salama, and M. Squire, "Telephony routing over IP (TRIP)," Internet Draft, Internet Engineering Task Force, Jan. 2000. Work in progress.

[2] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: session initiation protocol," Request for Comments 2543, Internet Engineering Task Force, Mar. 1999.

[3] J. Rosenberg and H. Schulzrinne, "A framework for telephony routing over ip," Request for Comments 2871, Internet Engineering Task Force, June 2000.

[4] International Telecommunication Union, "Packet based multimedia communication systems," Recommendation H.323, Telecommunication Standardization Sector of ITU, Geneva, Switzerland, Feb. 1998.