Abstract

Generic Router Assist (GRA) is a network-based service that enables end-to-end multicast transport protocols to take advantage of information distributed across the network elements in a given multicast distribution tree. The service consists of a canonical set of simple functions which network elements may apply to selected packets in the transport session as they traverse the distribution tree. The choice of function and the packet parameters to which it is applied can be defined and customized for a given transport session in a highly controlled fashion that still permits enough flexibility for GRA to be used to address a wide range of multicast transport problems not
amenable to end-to-end solution. This document provides the motivation and an architecture for GRA.
# Table of Contents

1. Introduction .................................................. 4  
2. Scope of Generic Router Assist ............................... 6  
3. Canonical Services and Functional Models/Examples ........ 11  
4. Implementation Considerations ................................. 15  
Abbreviations .................................................... 17  
References .......................................................... 18  
Revision History .................................................. 19  
Authors’ Addresses ................................................ 19
1. Introduction

The development of scalable end-to-end multicast protocols poses a tremendous challenge to network protocol designers. For example, the development of reliable multicast protocols has received considerable attention in recent years. Most protocols are based on an end-to-end solution [SRM, RMTP, TKP] and have found the problem of scaling to 1000s or even 100s of receivers daunting. The primary obstacles to the development of scalable protocols have been "feedback implosion" and "transmission isolation". The first of these concerns the difficulty of a large multicast application to limit feedback from receivers to a data source or to each other. The second concerns the difficulty of limiting the transmission of data to the subset of a multicast group that requires it.

Several proposals have been made to add functionality to routers for the purpose of improving the performance of multicast applications, particularly reliable multicast. Papadopoulos and Parulkar [LSM] introduced additional forwarding functionality to a router which would allow each router to identify a "special outgoing interface" over which to transmit a particular class of packets. They showed how this "turning point functionality" could be used to improve the performance of reliable multicast protocols. Levine and Garcia-Luna-Aceves [LABEL] proposed the addition of "routing labels" to routing tables which could be used to direct packets over specific interfaces. One of these, called a distance label, was shown to be quite useful in reliable multicast for directing requests for repairs to nearby repair servers. The third and, perhaps most relevant proposal is the PGM protocol [PGM]. Briefly, PGM is a reliable multicast protocol which uses negative acknowledgements (NAKs). The PGM protocol is an end-to-end transport protocol that contains a router component which performs NAK suppression and retransmission subcasting functionality. This proposal, like others [GMTS, BFS], is primarily motivated by PGM and the recognition of the benefits of exporting a set of flexible, simple router-based functionality for the purpose of multicast protocol design. Such functionality can significantly simplify the design of a large class of scalable multicast transport protocols.

In this draft, we present Generic Router Assist (GRA) functionality intended to help protocol designers deal with the two problems of feedback implosion and transmission isolation. This functionality is designed to assist in the scaling of receiver feedback information and in providing subcasting for large multicast groups. It consists of simple filtering and aggregation functions that reside within routers.

Signaling protocols are used by hosts to set up and invoke this
Briefly, a data source first initializes one or more desired services on its multicast tree using GRA setup messages. The GRA-capable routers on the tree then aggregate feedback from receivers and/or isolate transmissions through the use of filters set by either the sender or the receivers. For robustness, periodic transmissions of setup messages on the multicast tree are used to refresh GRA state in the face of routing changes and other possible errors. It should be stressed that GRA services are only invoked for certain packets; data packets are usually not treated any differently and will not cause any additional processing in routers.

GRA is not intended to provide sophisticated services which are difficult or impossible to implement in routers. GRA functionality is implemented at the IP layer and provides unreliable 'best-effort' services. Transport protocols which make use of GRA must be robust in the face of failures and the absence of GRA-capable routers in the network.

Before describing the details of GRA, we present a simple example in the context of a PGM-like reliable multicast protocol.

Consider a NAK-based reliable multicast protocol which places the responsibility of packet loss detection on each receiver. Each time that a receiver detects a loss (based on a gap in the sequence numbers of the packets that it receives), it unicasts a request for a repair (NAK) to the sender. Upon receipt of a NAK for a specific packet, the sender retransmits the packet to all receivers.

This protocol faces considerable challenges in dealing with multiple NAKs for the same packet. First, there is the problem of the sender having to process many NAKs. Second, there is the problem of limiting the number of retransmissions to the same packet. GRA can be used to (partially) solve these two problems. Prior to the transfer of any data, the application sets up a NAK aggregation filter at each GRA-capable router using a setup message. This filter is set up to suppress NAKs generated for the same packet. In addition, the router maintains information regarding the interfaces over which it has received NAKs so that it can subcast the retransmission on the portion of the multicast tree that contains receivers requiring a retransmission of the packet.

In Figure 1, we show how GRA can be used to aggregate feedback information in a reliable multicast transport protocol. In this figure, a multicast source (Src 1) transmits to two receivers (Rec 1 and Rec 2). The data packets from Src 1 are treated as regular multicast packets and forwarded accordingly. On the link between router R1 and router R2, a data packet is lost. Assuming a NAK based reliable multicast protocol, this loss causes the receivers to each send a NAK to
the source for the packet that was lost. In the example, receivers invoke GRA to send the NAKs to the source. Router R2 treats these NAKs in a special manner, removing the redundant NAKs to the source. Therefore, only one NAK arrives at the source. We see from this example that only certain types of packets require additional processing at GRA routers and that the majority of end-to-end packets are forwarded according to normal multicast forwarding rules (i.e. without additional router processing).

```
Src 1
  ^  |  |
   |  |  data packet
   R1  ^  |
        V
R2 suppresses duplicate +-----R2 <-----+
   NAK  |  ---  ---  |
   |  |  |  |  |
   |  |  |  |
   R3 R4
GRA  ^  |  ^  GRA
   NAK  |  |  NAK
   |  |  |
   Rec 1 Rec 2
```

Figure 1. Example of network support for transport protocols.

We conclude the introduction by commenting on the relationship of GRA to the multicast routing algorithm. GRA works on all types of multicast forwarding trees. However, GRA state is per session state and requires per session state in routers. If a source-based tree routing protocol is used to forward multicast packets, then this per session state will already exist in routers in the form of multicast forwarding entries. If a shared tree type multicast routing protocol is being used, then GRA per session state must be maintained on the shared tree. This is simply because GRA provides per session functionality.

2. Scope of Generic Router Assist

The types of services implemented with GRA are bounded by constraints and limitations of routing devices. In this section we explicitly describe the limitations and constraints of routing devices and
explain what we believe to be reasonable services to implement in routers.

We specifically describe router limitations in order to limit the scope of GRA. We believe that a small set of GRA services in routers can assist in scaling many of the problems in end-to-end multicast. Previous proposals have proposed complex elements that reside in routers to provide sophisticated capabilities. We feel that these are unreasonable for current generation routers. The approach of GRA is to provide a simple fixed set of services which give the maximum benefit with the least cost.

2.1. Service Properties

GRA services are performed on subset of packets sent between end-to-end transport protocols on the multicast distribution tree between a GRA Source and the set of GRA receivers. Only routers on the distribution tree for a particular GRA source act upon GRA packets. The advantages of GRA type services can only be realized when the actions are performed on packets that are directly on the forwarding tree of a multicast group.

In order to describe the requirements of GRA, we first describe the properties of what we feel are appropriate services.

   Fixed: by fixed services we mean those of which are statically part of router software or hardware. We DO NOT mean dynamically loadable modules. We feel that a fixed set of simple services will suffice for most of the scaling issues in transport protocols.

   Simple: We wish only to include those services that we feel are reasonable to be implemented in routers. These are services which can be performed with minimum CPU and memory overhead.

   Short Term: We wish to provide services for which state and processing overhead is short lived. GRA makes use of soft-state design principles.

2.2. GRA Requirements

When considering the service and architecture of GRA, we adhere to the following principles:

1. GRA services should be simple and fixed. They should not require excessive processing in routing devices nor should they buffer messages.
2. GRA services are not substitutes for well-engineered end-to-end protocol designs. We support the end-to-end design principles of transport protocols. GRA is an "assist" service which is designed to assist protocols in scaling aspects.

3. GRA services will not take on active networking attributes such as dynamically uploadable modules or programming language proposals.

4. GRA services should not directly participate in transport protocols. GRA should not be required for any transport protocols nor should any GRA services directly support a particular protocol.

5. GRA services should be those which may assist all or a reasonable subset of transport protocols.

6. GRA services should be used for assisting in *control* packet operations. GRA services should not be for the majority of packets in a multicast group.

2.3. Constraints of networking devices

Current generation routers perform processing of packets and execute routing and signaling protocols. Routers perform fast packet forwarding on the "forwarding path". This is usually hardware and software specifically designed for the forwarding of packets (and may include other functions such as policing, shaping, etc). This is in contrast to the "control plane" of a router where control protocols are run under an operating systems. Examples of these types of protocols are routing, management and signaling protocols.

We now describe the role and impact of GRA services on these two "planes" of a router.

Forwarding path: A router forwarding path usually consists of specialized hardware and software which is designed specifically for the purpose of forwarding packets. Newer routers also have abilites to perform other actions such as marking or policing for services such as QoS. In general, the forwarding path of routers is very limited in the amount of state and complexity of processing that can be performed. Although we do not rule out GRA services being implemented in the forwarding path of routers, we do not see this as feasible at this point in time. This is because of the more complicated processing rules for GRA packets (in comparison with basic forwarding operations) and the amount of state that is sometimes involved in performing GRA operations.
Control plane: Router control planes run embedded or general operating systems. On top of these operating systems are implementations of IP control protocols such as routing, signaling, and network management. The processing power and memory of a control plane is highly dependent on the hardware design of the router. Most routers use general purpose CISC or RISC processors for running the control plane operating system and protocols. The control plane is limited in processing by its hardware and the load generated from the other processes or tasks running. We expect that most GRA operations will be performed in the control plane where state and processing power are more readily available. However, we do stress that routers generally have fairly slow control plane hardware. This is to keep the cost low and because the processing required for control plane protocols is usually low.

2.4. State Constraints

As we evaluate particular services which are candidates for inclusion in GRA, constraints in router state are an important consideration. We wish to select services which will not create substantial or long-lived state.

2.4.1. Session State

Routers which perform multicast forwarding contain per tree forwarding state. For trees rooted at multicast sources, this amounts to per source per group forwarding entries. This state must be kept in both the forwarding as well as the control plane. GRA services are per session, or per source. This is simply because GRA services are per transport session.

The session state of GRA does not impose much additional state to routers. Per each session, a state block describing the services desired is required. In addition, other types of state may be created in the course of a GRA session. The GRA session state is the only state required for the length of a session; other state is setup and torn down in smaller intervals.

2.4.2. Packet State

The implementation of particular services may cause per packet state in GRA routers. Services which cause per packet state should use small timers for tearing down this state, which could explode in certain conditions. An example of a service which causes per packet state is NAK elimination. We feel the use of small timers will minimize problems in state growth.
2.4.3. Buffering

As part of the requirements for GRA services, we specify that the buffering of packets is unacceptable. GRA services may drop or modify packets in transit, but they should never buffer packets.

We feel that the buffering of packets in routing devices is unacceptable due to the unpredictable behavior of such a service. In addition we feel that this is an unreasonable service for routers to support without a significant payback in end-to-end scaling.

2.5. Processing Constraints

GRA services may require differing amount of computation. As a general rule, GRA services should require minimal computation and packet manipulation.

2.5.1. Computation

We expect most GRA services to require minimal computation. Services which require minimal computation are reasonable to implement in routing devices and minimize security risk. Examples of operations which we feel are appropriate are:

Comparison: comparison operations may be performed on GRA packets against the GRA session state in the router in order to determine whether a service should be invoked. Examples of comparison operations are: equal to, less than, greater than, etc.

Basic Predicates: When receiving a GRA packet, a router may be required to update its GRA session state. The operations required for updating the state should remain simple. Example of simple operations are addition, subtraction, etc.

In order to limit the computational complexity of GRA services, we recommend that GRA operations remain singular. The combination of predicates and operations creates problems in both router processing and in GRA specifications themselves. We wish to avoid problems in operator and action precedence.

2.5.2. Buffer Operations

The possibility exists to define services which require extensive packet manipulation by GRA routers. We feel that extensive manipulation of packets is expensive and therefore unreasonable for routers. GRA services should be able to be invoked without extensive manipulation of packets. Services which update or overwrite fields are acceptable. Services which require the formation of new packets or
aggregate information into new packets are unacceptable.

2.6. Examples of "Reasonable" Services

In this section we briefly describe three GRA services which we feel are appropriate and why we feel they meet the above requirements. The two example services are those that provide general functions which do not incur large amounts of state or processing.

Elimination: Elimination is the selected dropping of redundant signaling. An example of elimination is NAK suppression in a reliable multicast protocol. Elimination is a service which requires little computational overhead. It does however, require per packet (or per sequence block) state. This state can be controlled by the use of short soft state timers.

Subcasting: Subcasting is the forwarding of a multicast packet to a subset of the multicast tree. Subcasting is useful for a variety of multicast protocols. Subcasting does not require a large amount of processing. The state required is an identifier for the subtree and a list of outgoing interfaces. This state is similar to multicast forwarding tree state.

3. Canonical Services and Functional Models/Examples

While a variety of mechanisms must come together to enable a specific GRA service in a distribution tree (session path messages to establish session parameters and neighbour information, a control protocol to define, enable, and disable specific filtering services, etc.), the basic mechanism of GRA consists of router based services that can be described in the language of filters, keys, and conditional functions (binary predicates and their outcomes).

Based upon the examples (of predicate elimination and sub-casting) in the previous sections, this section expands the model and terminology of filters to describe the full flexibility of GRA. The intent here is to generalize the mechanism fully enough to be able to accommodate currently unspecified requirements for multicast transport services as they emerge.

Revisiting the example of predicate elimination, note that the receipt of a loss report at a router implies several things. The packet type implies that a type of filter should be established for the transport session, that the filter key is (a sequence number) of a particular length at a particular offset, that the loss report in hand should be forwarded, that the interface upon which the loss report was received along with the count of parity packets requested should be recorded and associated with the key in the filter, and
that subsequent matching loss reports on any interface should be eliminated unless they request a larger number of parity packets than has been requested on any interface. The filter itself has other implied characteristics. It eliminates only for a certain interval after forwarding the initiating or any subsequent loss report, and it has an implied lifetime after which it is locally expired by the router.

Similarly in the example of sub-casting, the receipt of a retransmission at a router implies several things. The packet type implies that the packet should be matched against an existing filter type based on a key of a particular length and at a particular offset, that the packet itself should be forwarded on all interfaces with non-zero counts recorded as state associated with the key, that the corresponding count should be decremented by 1 per interface until it reaches 0, and that the key’s state should be discarded once all counts are zero. By implication, unmatched retransmissions should not be forwarded.

From these examples some generalities emerge which, once they are extricated from their specific semantics, can be re-assembled in a variety of useful ways to provide router-based assistance for a broader class of transport services.

3.1. General Model

The general model is one in which packets carry one of a tightly constrained set of signals that alert the router to apply a pre-defined filtering service.

The packet-borne signal conveys

- a filter type,
- an associated transaction,
- a key,
- and possible packet variables
  (the parity count in the case of parity loss reports).

Canonical filtering services in routers can be defined by

- a filter type,
- associated supported transactions,
- predicated on specific conditions,
- and three functions gated on that predicate whether TRUE or FALSE:

  - \( f(p) \): how to dispose of the packet,
  - \( f(s) \): how to transform the key’s state, and
  - \( f(v) \): how to transform the outgoing interface (OIF) list
associated with the key.

All of these as well as the offsets and lengths of the key and any packet variables for each supported transaction constitute the definition of the filtering service.

3.2. An Example Using the General Model

Given this model, the handling of parity NAKs and parity retransmissions in PGM can be described as a predicate eliminating and subcasting filter.

Let IIF be the interface on which a packet is received. Let COUNT be the number of parity packets requested and recorded either in the loss report (RQST COUNT), as interface state (OIF COUNT), or as key state (HIGH COUNT). Let ET be the elimination timer for a particular key’s state, and LT be the life timer for a particular key’s state.

The filtering service in the router supports two transactions, one inbound (RCVR_UPDATE) and one outbound (FORWARD).

3.2.1. RCVR_UPDATE

For RCVR_UPDATE, in addition to a transport session identifier, the following are defined for the signal in the packet:

- ELIM_SUBC is the filter type
- RCVR_UPDATE is the transaction
- SQN is the key
- COUNT is a packet-borne variable

For RCVR_UPDATE, the following are defined for the filtering service in the router:

In case the key fails to match existing key state:

predicate: NOOP
f(v): OIF COUNT for IIF = MAX(RQST COUNT, 0)
f(s): start ET, start LT, HIGH COUNT = RQST COUNT
f(p): reverse forward to upstream neighbour

In case the key matches existing key state:

predicate: ET is running or RQST COUNT LEQ HIGH COUNT

In case the predicate is TRUE:

f(v): OIF COUNT for IIF = MAX(RQST COUNT, OIF COUNT for IIF)
f(s): restart LT
f(p): discard

In case the predicate is FALSE:

f(v): OIF COUNT for IIF = MAX(RQST COUNT, OIF COUNT for IIF)
f(s): restart ET, HIGH COUNT = RQST COUNT
f(p): reverse forward to upstream neighbour

3.2.2. FORWARD

For FORWARD, in addition to a transport session identifier, the fol-
lowing are defined for the signal in the packet:

ELIM_SUBC   is the filter type
FORWARD     is the transaction
SQN         is the key

For FORWARD, the following are defined for the filtering service in
the router:

In case the key fails to match existing key state:

predicate: NOOP
f(v):      NOOP
f(s):      NOOP
f(p):      discard

In case the key matches existing key state:

predicate: for all OIF COUNTs, OIF COUNT NE 0

In case the predicate is TRUE:

f(v): decrement OIF COUNT
f(s): NOOP
f(p): forward on OIF

In case the predicate is FALSE:

f(v): NOOP
f(s): NOOP
f(p): discard

Associated with the filtering service would be an additional house-
keeping function which would discard a key’s state if either LT
expired or all the OIF COUNTs were zero.
The point of this example (eventually, "these examples") is to demonstrate that this model for GRA can accommodate a highly functional set of router-based services which, given a transport-layer-independent implementation, could be provided by routers for general deployment by transport protocols engineered to signal the routers in a generic way. Now that we have established the context and a model for GRA, the next section discusses implementation considerations in the interest of making the mechanisms of GRA more concrete, and highlighting their practical and performance consequences to traditional multicast forwarding.

4. Implementation Considerations

4.1. Signalling Protocol - GRA service indicators

A consideration of the implementation issues attending GRA strongly determines the class of functions that may be realized with this mechanism. These considerations relate to both security of the router and performance in the forwarding path. The former will be dealt with in another section. This section outlines the time and space scaling consequences of GRA to the forwarding path.

To be a generic network layer service, it’s clear that some minimal indicator is required in the network layer to signal the presence of GRA signal on a packet. As has been previously noted, remember that the GRA indicator is typically NOT borne by basic data packets. These are switched without exception in the usual forwarding path. In this section, packets bearing a GRA indicator will be referred to as GRA packets.

A network layer indicator frees forwarding engines from the burden of having to walk into and parse transport headers on every switched packet. It’s highly efficient to encode the GRA indicator on the network layer header since that header is typically already within the grasp of the forwarding engine. In PGM, this indicator is implemented with an IP Router Alert option, but a single bit in the basic IP header would function just as well (even better, actually, for legacy routers).

Once GRA packets can be detected in the forwarding path, the next step is to locate and parse the GRA parameters: the filter type, the transaction, the key, and the variables from above. These should be encoded as TLVs located somewhere between the end of the network layer header and the beginning of the transport layer payload or SDU.

It’s tempting in the case of the key and the packet variables to consider encoding their offsets and lengths in a TLV that could be used to locate the actual values themselves in the transport header or,
more wildly, in the SDU, but this would amount to providing a near 
programmatic directive to the network element which is fraught with 
risk. So note that, in the example of elimination above, the number 
of loss reports requested would, in this model, be encoded both in 
its natural place in the transport header, and also as a GRA-specific 
variable in a separate GRA TLV.

An alternative is to establish the location and length of keys and 
packet variables as attributes of the filtering service defined in 
the network element itself so that the only vulnerability to the net-
work elements would derive from abuse of the setup protocol in the 
control plane in place of vulnerability in the forwarding path or 
data plane.

The location of GRA TLVs before, inside, or after the transport layer 
header is at the crux of whether GRA is regarded as integral to a 
specific layer or as a shim layer between layers. The answer to the 
question determines not just where to locate the TLVs but also where 
to implement the service in host protocol stacks.

As for forwarding-time implementation of GRA services, we take it as 
a requirement for this specification that some services must be 
light-weight enough to be candidates for optimized implementations in 
hardware or firmware, while others may be sufficiently complex to 
warrant software implementations. Soft-state-based services that do 
not maintain per-packet state may be in the first class; services 
that maintain per-packet state and therefor require a 
sorted/searchable data structure may be in the second class.

Beyond these very GRA-specific issues are the more general control 
protocol issues of how to interoperate network elements with 
disparate GRA capabilities, and all of the specifics of defining, 
enabling, and disabling filters themselves. These issues are best 
treated once a solid model of the GRA mechanism itself is esta-
blished.

4.2. Control Protocol - Filter definition, enabling, and disabling

4.3. Control Protocol - Session path messages and neighbour informa-
tion
Abbreviations

IIF    Incoming Interface
NAK    Negative Acknowledgement
NOOP   No Operation
OIF    Outgoing Interface
SDU    Service Data Unit
SQN    Sequence Number
TLV    Type Length Value
References


Revision History

draft-ietf-rmt-gra-00.txt October 1999

Original draft.

Authors’ Addresses

Brad Cain
bcain@nortelnetworks.com

Tony Speakman
speakman@cisco.com

Don Towsley
towsley@cs.umass.edu