Network Working Group
Internet Draft
Expires: July 2007

J.Halpern
Self
Huaiyuan Ma
Huawei Technologies Co., Ltd

January 16, 2007

A VPN Library for use with the ForCES Protocol and Model
draft-halpern-forces-lfblibrary-vpn--00

Status of this Memo

Abstract

The forwarding and Control Element Separation (ForCES) protocol defines a standard communication and control mechanism through which a Control Element (CE) can control the behavior of a Forwarding Element (FE). That control is accomplished through manipulating

attributes of Logical Function Blocks (LFBs), whose structure is defined in a model RFC produced by the working group. In order to build an actual solution based on this protocol, defining a set of Logical Function Block definitions that can be instantiated by FEs and controlled by CEs is welcome. A base library definition of LFBs is already given in library [5]. VPN (Virtual Private Network) services, as a kind of important services widely employed in Internet, will certainly be implemented in routers using this protocol. This document provides an initial set of VPN LFB definitions in particular, a set of tunnel encapsulator and decapsulator LFBs. It is anticipated that additional VPN-related LFB definitions like L2VPN, L3VPN can be defined over time.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119.

Table of Contents

1. Introduction

In a solution using ForCES protocol, Control Elements (CEs) can control the behavior of Forwarding Elements (FEs) through manipulating attributes of Logical Function Blocks (LFBs). LFB's structure and abstract semantics is defined in Model [6]. That document also defines a single LFB Class for gaining access to FE properties including the set of LFBs and their interconnection. A LFB class is defined to manipulate the protocol properties of the FE in the protocol [4].

In I-D draft [5], a set of LFBs which are necessary to implement basic forwarding process are defined. This draft is intended to define an initial set of LFBs for a kind of important Internet service, Virtual Private Network (VPN). It is expected that other VPN-related definitions will be developed over time.

2. Tunnel Definitions

2.1. GRE Tunnel Definitions

GRE tunnel specification and its extension are described in RFC 2784 and RFC 2890 respectively. A GRE tunnel is composed of three parts: outer delivery header, followed by a GRE header which is followed by a payload packet. The format of outer delivery header is determined by the corresponding delivery protocol, IPv4 or IPv6. In GRE header, there are two important optional fields: one is called GRE key which is intended to be used for identifying an individual traffic flow within a tunnel; the other is called Sequence Number, the Sequence Number MUST be used by the receiver to establish the order in which packets have been transmitted from the encapsulator to the receiver. The intended use of the Sequence Field is to provide unreliable but in-order delivery. The payload packet would be IPv4, IPv6 etc.

When forwarding an IP packet in a next-hop applicator LFB, the matched FIB entry indicates the packet should be transmitted over a GRE tunnel and a correct tunnel index can be found out in the FIB entry. The original packet will be fed to GRE tunnel encapsulator with tunnel index as meta-data together in the downstream forwarding process. Tunnel index points to the correct tunnel entry in tunnel configuration table which stores the information of each tunnel. Each

tunnel entry maintains a management flag called "TunnelState" indicating whether the current tunnel is administratively up.

GRE tunnel maintains a fragmentation permitted flag. Before encapsulating a payload packet GRE tunnel encapsulator will check the size of that payload packet against MTU. If the size of tunnelled packet is larger than MTU and at this moment that fragmentation permitted flag should be checked, if that flag is set, then chop the original packet into small packets with appropriate size, otherwise, send ICMP message to notify the appropriate receiver of some error. If the delivery protocol is IPv4, then the format of outer delivery header would be IPv4, otherwise, it would be IPv6.

Once a IP packet with GRE is produced from an output port from GRE tunnel encapsulator, a meta-data called "NextHopReference" which points to the index of correct FIB entry is accompanied at the same time, that is, an alias entry pointing to the next-hop table so that it can use the predetermined route to the tunnel end-point.

When a classifier LFB identifies the incoming packet is an IP packet with GRE at the GRE tunnel exit point, it will feed that packet to the GRE tunnel decapsulator, which will find out the correct tunnel index which maintains a local VPN ID field, the GRE tunnel decapsulator then strips off the outer delivery header and GRE header and feed it to the forwarding-related LFB with local index ID as meta-data indicating which VPN that payload packet belongs to.

The actual GRE tunnel LFB class is defined as below.

```
<LFBLibrary provides="GRE_Tunnel_LFB">
  <load library="Base"/>
  <dataTypeDefs>
    <dataTypeDef>
    <name>NextHopIndex</name>
    <synopsis>
      An index used by the next hop table.
      Typically stored in and generated as metadata by
      the longest-prefix-match LFB
    </synopsis>
    <typeRef>int32</typeRef>
```

```
       </dataTypeDef>
       <dataTypeDef>
         <name>VPNID</name>
         <synopsis>
             An ID used to provide context for
             VPN specific Packet processing.
         </synopsis>
         <typeRef>int32</typeRef>
       </dataTypeDef>
       <dataTypeDef>
         <name>GRETunnelTableType</name>
         <synopsis>
            GRE tunnel configuration table
            Each table entry describes a single GRE tunnel.
            The table Index is input meta-data for the encapsulator.
         </synopsis>
         <array type="variable-size">
         <struct>
           <element elementID="1">
             <name>TunnelValid</name>
             <synopsis>
                It is enabled or disabled by FIB indicating whether the current
tunnel is permitted to
                be used in forwarding process.
             </synopsis>
             <typeRef>boolean</typeRef>
           </element>
           <element elementID="2">
             <name>FragmentationPermitted</name>
             <synopsis>
               it indicates whether it permits fragmentation when the size of a
packet exceeds
               the tunnel's MTU.
             </synopsis>
             <typeRef>boolean</typeRef>
           </element>
           <element elementID="3">
             <name>ChecksumNeeded</name>
             <synopsis>
               it indicates whether a checksum is needed.
             </synopsis>
             <typeRef>boolean</typeRef>
           </element>
           <element elementID="4">
```

```
      <name>PacketType</name>
                  ...needs definition ...
            ... probably for decapsulator error checking? ...
    </element>
    <element elementID="5">
      <name>SrcAddr</name>
      <synopsis>
         IP Address for local end of tunnel
      </synopsis>
      <typeRef>IPAddress</typeRef>
    </element>
    <element elementID="6">
      <name>DstAddr</name>
      <synopsis>
         Address for remote End of Tunnel
      </synopsis>
      <typeRef>IPAddress</typeRef>
    </element>
    <element elementID="7">
      <name>GREKey</name>
      <synopsis>
          Key for this specific GRE Tunnel
          The presence of this element indicate this tunnel uses
          keyed GRE format.
      </synopsis>
      <optional/>
      <typeRef>int32</typeRef>
    </element>
    <element elementID="8">
      <name>NextHopReference</name>
      <synopsis>
         Reference to the correct NextHopIndex
         This points to a LPM where the next hop is maintained.
         The information is put in the encapsulator meta-data.
      </synopsis>
      <alias>NextHopIndex</alias>
    </element>
    <element elementID="9">
      <name>MTU</name>
      <synopsis>
           Maximum Transmit Unit Used in conjunction with the flags
           to decide if large packets should be encapsulated, fragmented,
           or errored.
      </synopsis>
      <typeRef>uint32</typeRef>
    </element>
    <element elementID="10">
```

```
            <name>LocalVPNID</name>
            <synopsis>
                VPN ID used by decapsulator as generated
                meta-data
            </synopsis>
            <typeRef>VPNID</typeRef>
          </element>
        <element elementID="11">
            <name>TunnelState</name>
            <synopsis>
                It indicates whether the current tunnel is administratively up
or down.
            </synopsis>
            <typeRef>Boolean</typeRef>
          </element>
        <element elementID="12">
            <name>SequencingNeeded</name>
            <synopsis>
              it indicates whether a sequence field to differentiate different
traffic flows in a
               tunnel is needed.
            </synopsis>
            <typeRef>boolean</typeRef>
        </element>
        <element elementID="13">
            <name>SequenceNumber</name>
            <synopsis>
              a sequence field to differentiate different traffic flows in a
tunnel.
            </synopsis>
            <optional/>
            <typeRef>int32</typeRef>
        </element>
        <element elementID="14">
            <name>Checksum</name>
            <synopsis>
              The Checksum field contains the IP (one's complement) checksum
sum of the all the 16 bit
              words in the GRE header and the payload packet.
            </synopsis>
            <optional/>
            <typeRef>int16</typeRef>
        </element>
        </struct>
        </array>
      </dataTypeDef>
    </dataTypeDefs>
```

```
     <LFBClassDefs>
       <LFBClassDef LFBClassID="0x00010010">
         <name>GRE_Encapsulator</name>
         <synopsis>
           It specifies how to encapsulate an IP packet so that it can be
transmitted over GRE tunnel.
         </synopsis>
         <version>0.0</version>
         <inputPorts>
           <inputPort>
             <name>PacketIn</name>
             <synopsis>
               Normal packet in.
             </synopsis>
             <expectation>
               <frameExpected>
                 <ref>IPv4</ref>
                 <ref>IPv6</ref>
               </frameExpected>
               <metadataExpected>
                 <ref>Tunnel_Index</ref>
               </metadataExpected>
             </expectation>
           </inputPort>
         </inputPorts>
         <outputPorts>
            <outputPort>
              <name>PacketOut</name>
              <synopsis>
                IP packet with GRE
              </synopsis>
              <product>
                <frameProduced>
                 <ref> GREFrame </ref>
                </frameProduced>
                <metadataProduced>
                  <ref>NextHopReference</ref>
                </metadataProduced>
              </product>
             </outputPort>
         <outputPort>
              <name>FailOutput</name>
              <synopsis>
                error prompt information to indicate whether some operations
like fragmentation are
```

```
            permitted.
              </synopsis>
              <product>
                <frameProduced>
                  <ref>taggedFrame</ref>
                </frameProduced>
                <metadataProduced>
                  <ref>errorid</ref>
                </metadataProduced>
              </product>
             </outputPort>
          </outputPorts>
          <attributes>
              <attribute access="read-write" elementID="1">
                  <name>GRETunnelTable</name>
                  <synopsis>
                      Table of GRE Tunnels supported by this
                      encapsulator
                  </synopsis>
              <typeRef>GRETunnelTableType</typeRef>
              </attribute>
          </attributes>
          <description>
            when fowarding a IP packet, a matching FIB entry has a GRE tunnel
flag which indicates it
            should be transmitted over a GRE tunnel, then according to the
constrains, TTL, MTU,
            fragmentation flag etc in the GRE tunnel entry to encapsulate a IP
packet in its payload
            part.
          </description>
         </LFBClassDef>
         <LFBClassDef LFBClassID="0x00010011">
          <name>GRE_Decapsulator</name>
          <synopsis>
            It specifies the procedure how to decapsulate a tunnelled packet.
          </synopsis>
          <version>0.0</version>
          <inputPorts>
            <inputPort>
              <name>PacketIn</name>
              <synopsis>
                A IP packet with GRE.
              </synopsis>
              <expectation>
```

```
              <frameExpected>
                <ref>GREFrame</ref>
              </frameExpected>
            </expectation>
          </inputPort>
        </inputPorts>
        <outputPorts>
          <outputPort>
            <name>PacketOut</name>
            <synopsis>
              original packet output
            </synopsis>
            <product>
              <frameProduced>
                <ref>IPv4</ref>
                <ref>IPv6</ref>
              </frameProduced>
              <metadataProduced>
                 <ref>LocalVPNID</ref>
              </metadataProduced>
            </product>
          </outputPort>
          <outputPort>
             <name> FailOutput </name>
             <synopsis>
                error prompt information generated when a GRE packet cannot
match a GRE tunnel
                state
              </synopsis>
              <product>
                <frameProduced>
                  <ref>taggedFrame</ref>
                </frameProduced>
                <metadataProduced>
                  <ref>errorid</ref>
                </metadataProduced>
              </product>
          </outputPort>
        </outputPorts>
        <attributes>
         <attribute access="read-write" elemendID="x">
         <name>GRETunnelTable</name>
         <synopsis>
              Reference to the GRE Tunnel Table this decapsulator uses
```

```
              which is stored in the corresponding encapsulator.
          </synopsis>
          <alias>GRETunelTableType</alias>
          </attribute>
         </attributes>
         <description>
           Once the classifier has determined the content of an incoming packet
is GRE, it will be fed
           to a GRE decapsulator, which can achieve the local VPN ID to which
the packet belongs and
           strip off the outer IP header and GRE shim, at the same time, treat
local VPN ID as meta-
           data and then feed it and original packet to the down-stream LFBs.
         </description>
       </LFBClassDef>
     </LFBClassDefs>
   </LFBLibrary>
```

3. Security Considerations

These definitions if used by an FE to support ForCES create manipulable entities on the FE, Manipulation of such objects can produce almost unlimited effects on the FE. FEs should ensure that only properly authenticated ForCES protocol participants are performing such manipulations. Thus, largely, the security issues with this protocol are defined in Protocol [2].

4. Acknowledgments

Thanks Zengjie,Kou for providing some comments.

5. References

5.1. Normative References

[1] Khosravi, et al. Requirements for Separation of IP Control and Forwarding, RFC 3654, November 2003.

[2] L. Yang, et al. ForCES Architectural Framework, RFC 3746, April 2004.

[3] Yang, L., Halpern, J., Gopal, R., DeKok, A., Haraszti, Z., and S. Blake, "ForCES Forwarding Element Model", Feb. 2005.

[4] A. Doria, et al. ForCES Protocol Specification, draft-ietf- forces-protocol-06.txt, December 2005.

[5] Joel M.Halpern, A base Library for use with the ForCES Protocol and Model, draft-halpern-forces-lfblibrary-base-01.txt, March, 2006

[6] L.Yang, et al. ForCES Forwarding Element Model, draft-ietf-forces-model-06.txt

5.2. Informative References

Author's Addresses

Joel M. Halpern
Self
P. O. Box 6049
Leesburg, VA 20178
US

Huanyuan Ma
Huawei Technologies Co., Ltd
mahuaiyuan@huawei.com

Zengjie Kou
Huawei Technologies Co., Ltd
kouzengjie@huawei.com

Intellectual Property Statement

on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.