DragonFlyBSD - Bug #633

ICMP extensions for MPLS support for traceroute(8)

05/10/2007 11:19 AM - hasso

Status:	Closed	**Start date:**	
Priority:	Low	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

HG changeset patch
User Hasso Tepper <hasso@estpak.ee>
Date 1178795442 -10800
Node ID da51ceae15a698a0d629dcbc0d3e834904e85069
Parent ff1a613aa7893b9abe16bdcb6844d0370b80147b
ICMP extensions for MPLS support for traceroute(8).

If noone objects, I will commit tomorrow.

Obtained-from: NetBSD

diff -r ff1a613aa789 -r da51ceae15a6 usr.sbin/traceroute/traceroute.8
--- a/usr.sbin/traceroute/traceroute.8 Wed May 09 23:30:51 2007 +0300
+++ b/usr.sbin/traceroute/traceroute.8 Thu May 10 14:10:42 2007 +0300
@@ -43,7 +43,7 @@
.Sh SYNOPSIS
.Nm traceroute
.Bk -words
-.Op Fl cdDIlnrSv
+.Op Fl cdDIlMnrSv
.Op Fl f Ar first_ttl
.Op Fl g Ar gateway_addr
.Op Fl m Ar max_ttl
@@ -109,6 +109,8 @@ The default is the value of the system's
The default is the value of the system's
.Cm net.inet.ip.ttl
MIB variable, which defaults to 64.
+.It Fl M
+If found, show the MPLS Label and the Experimental (EXP) bit for the hop.
.It Fl n
Print hop addresses numerically rather than symbolically and numerically
(saves a nameserver address-to-name lookup for each gateway found on the
diff -r ff1a613aa789 -r da51ceae15a6 usr.sbin/traceroute/traceroute.c
--- a/usr.sbin/traceroute/traceroute.c Wed May 09 23:30:51 2007 +0300
+++ b/usr.sbin/traceroute/traceroute.c Thu May 10 14:10:42 2007 +0300
@@ -244,6 +244,54 @@ struct packetdata {
u_int32_t usec;
};

+/*
+ * Support for ICMP extensions
+ *
+ * http://www.ietf.org/proceedings/01aug/I-D/draft-ietf-mpls-icmp-02.txt
+ */
+#define ICMP_EXT_OFFSET 8 + 128 /* ICMP type, code, checksum (unused)
+   * + original datagram */
+#define ICMP_EXT_VERSION 2
+/*
+ * ICMP extensions, common header
+ */
+struct icmp_ext_cmn_hdr {
+#if BYTE_ORDER == BIG_ENDIAN

```
+ unsigned char   version:4;
+ unsigned char   reserved1:4;
+#else
+ unsigned char   reserved1:4;
+ unsigned char   version:4;
+#endif
+ unsigned char   reserved2;
+ unsigned short  checksum;
+};
+
+/*
+ * ICMP extensions, object header
+ */
+struct icmp_ext_obj_hdr {
+ u_short length;
+ u_char  class_num;
+#define MPLS_STACK_ENTRY_CLASS 1
+ u_char  c_type;
+#define MPLS_STACK_ENTRY_C_TYPE 1
+};
+
+struct mpls_header {
+#if BYTE_ORDER == BIG_ENDIAN
+ uint32_t       label:20;
+ unsigned char  exp:3;
+ unsigned char  s:1;
+ unsigned char  ttl:8;
+#else
+ unsigned char  ttl:8;
+ unsigned char  s:1;
+ unsigned char  exp:3;
+ uint32_t       label:20;
+#endif
+};
+
struct in_addr gateway[MAX_LSRR + 1];
int lsrrlen = 0;
int32_t sec_perturb;
@@ -251,6 +299,7 @@ int32_t usec_perturb;

u_char packet[512], *outpacket; /* last inbound (icmp) packet */

+void decode_extensions(unsigned char *, int);
void dump_packet(void);
int wait_for_reply(int, struct sockaddr_in *, struct timeval *);
void send_probe(int, u_int8_t, int, struct sockaddr_in *);
@@ -283,6 +332,7 @@ int waittime = 5;  /* time to wait for r
int waittime = 5;  /* time to wait for response (in seconds) */
int nflag;   /* print addresses numerically */
int dump;
+int Mflag;   /* show MPLS labels if any */

int
main(int argc, char *argv[])
@@ -310,7 +360,7 @@ main(int argc, char *argv[])

sysctl(mib, sizeof(mib)/sizeof(mib[0]), &max_ttl, &size, NULL, 0);

- while ((ch = getopt(argc, argv, "SDIdg:f:m:np:q:rs:t:w:vIP:c")) != -1)
+ while ((ch = getopt(argc, argv, "SDIdg:f:m:np:q:rs:t:w:vIP:cM")) != -1)
switch (ch) {
case 'S':
sump = 1;
@@ -363,6 +413,9 @@ main(int argc, char *argv[])
errx(1, "max ttl must be %u to %u.", first_ttl,
MAXTTL);
max_ttl = (u_int8_t)l;
```

```
+   break;
+  case 'M':
+   Mflag = 1;
break;
case 'n':
nflag++;
@@ -681,6 +734,8 @@ main(int argc, char *argv[])
timeout++;
loss++;
}
+   else if (cc && probe == nprobes - 1 && Mflag)
+    decode_extensions(packet, cc);
fflush(stdout);
}
if (sump)
@@ -721,6 +776,118 @@ wait_for_reply(int sock, struct sockaddr

free(fdsp);
return (cc);
+}
+
+void
+decode_extensions(unsigned char *buf, int ip_len)
+{
+  struct icmp_ext_cmn_hdr *cmn_hdr;
+  struct icmp_ext_obj_hdr *obj_hdr;
+  union {
+   struct mpls_header mpls;
+   uint32_t mpls_h;
+  } mpls;
+  int data_len, obj_len;
+  struct ip *ip;
+
+  ip = (struct ip *)buf;
+
+  if (ip_len <= (int)(sizeof(struct ip) + ICMP_EXT_OFFSET)) {
+  /*
+   * No support for ICMP extensions on this host
+   */
+  return;
+  }
+
+  /*
+   * Move forward to the start of the ICMP extensions, if present
+   */
+  buf += (ip->ip_hl << 2) + ICMP_EXT_OFFSET;
+  cmn_hdr = (struct icmp_ext_cmn_hdr *)buf;
+
+  if (cmn_hdr->version != ICMP_EXT_VERSION) {
+  /*
+   * Unknown version
+   */
+  return;
+  }
+
+  data_len = ip_len - ((u_char *)cmn_hdr - (u_char *)ip);
+
+  /*
+   * Check the checksum, cmn_hdr->checksum == 0 means no checksum'ing
+   * done by sender.
+   *
+   * If the checksum is ok, we'll get 0, as the checksum is calculated
+   * with the checksum field being 0'd.
+   */
+  if (ntohs(cmn_hdr->checksum) &&
+      in_cksum((u_short *)cmn_hdr, data_len)) {
+  return;
```

```
+ }
+
+ buf += sizeof(*cmn_hdr);
+ data_len -= sizeof(*cmn_hdr);
+
+ while (data_len > 0) {
+ obj_hdr = (struct icmp_ext_obj_hdr *)buf;
+ obj_len = ntohs(obj_hdr->length);
+
+ /*
+  * Sanity check the length field
+  */
+ if (obj_len > data_len) {
+  return;
+ }
+
+ data_len -= obj_len;
+
+ /*
+  * Move past the object header
+  */
+ buf += sizeof(struct icmp_ext_obj_hdr);
+ obj_len -= sizeof(struct icmp_ext_obj_hdr);
+
+ switch (obj_hdr->class_num) {
+ case MPLS_STACK_ENTRY_CLASS:
+  switch (obj_hdr->c_type) {
+  case MPLS_STACK_ENTRY_C_TYPE:
+   while (obj_len >= (int)sizeof(uint32_t)) {
+    mpls.mpls_h = ntohl(*(uint32_t *)buf);
+
+    buf += sizeof(uint32_t);
+    obj_len -= sizeof(uint32_t);
+    printf(" [MPLS: Label %d Exp %d]",
+        mpls.mpls.label, mpls.mpls.exp);
+   }
+   if (obj_len > 0) {
+    /*
+     * Something went wrong, and we're at
+     * a unknown offset into the packet,
+     * ditch the rest of it.
+     */
+    return;
+   }
+   break;
+  default:
+   /*
+    * Unknown object, skip past it
+    */
+   buf += ntohs(obj_hdr->length) -
+       sizeof(struct icmp_ext_obj_hdr);
+   break;
+  }
+  break;
+
+ default:
+  /*
+   * Unknown object, skip past it
+   */
+  buf += ntohs(obj_hdr->length) -
+      sizeof(struct icmp_ext_obj_hdr);
+  break;
+ }
+ }
}

void
```

```
@@ -1022,7 +1189,7 @@ usage(void)
usage(void)
{
fprintf(stderr,
-     "usage: %s [-cdDIlnrSv] [-f first_ttl] [-g gateway_addr] [-m max_ttl]\n"
+     "usage: %s [-cdDIlMnrSv] [-f first_ttl] [-g gateway_addr] [-m max_ttl]\n"
"\t[-p port] [-P proto] [-q nqueries] [-s src_addr] [-t tos]\n"
"\t[-w waittime] host [packetsize]\n", getprogname());
exit(1);
```

History

#1 - 05/10/2007 12:49 PM - joerg

Please don't add new code that depends on the order of bitfields.
(Or bitfields in general for wire protocols).

Joerg

#2 - 05/10/2007 07:20 PM - dillon

```
:> +#if BYTE_ORDER == BIG_ENDIAN
:> + unsigned char   version:4;
:> + unsigned char   reserved1:4;
:> +#else
:> + unsigned char   reserved1:4;
:> + unsigned char   version:4;
:> +#endif
:> + unsigned char   reserved2;
:> + unsigned short  checksum;
:> +};
:
:Please don't add new code that depends on the order of bitfields.
:(Or bitfields in general for wire protocols).
:
:Joerg
```

What Joerg is saying is that the patch looks great! But we're
trying to get away from using bitfields so if it would not be too
much trouble, could you just make that a u_char and extract the fields
manually with a macro?

Note that when used as a u_char, I'm pretty sure that no endian
conversions are needed when doing a manual extraction. Endian
conversions on bitfield ordering within single bytes are an artifact
of the way the compiler assigns the bitfield indices. The actual
storage (within a char) is the same.

Take for example the IP header (now there's something I would like to
clean up and just make _IP_VHL the default). Note that the IP_VHL_*()
macros do not require any endian conditionalization.

```
struct ip {
#ifdef _IP_VHL
u_char  ip_vhl;                 /* version << 4 | header length >> 2 */
#elif _BYTE_ORDER == _LITTLE_ENDIAN
u_int   ip_hl:4,                /* header length */
ip_v:4;                 /* version */
#elif _BYTE_ORDER == _BIG_ENDIAN
u_int   ip_v:4,                 /* version */
ip_hl:4;                /* header length */
#else
...
#endif

#define  IP_VHL_HL(vhl)         ((vhl) & 0x0f)
#define  IP_VHL_V(vhl)          ((vhl) >> 4)
#define  IP_VHL_BORING          0x45
```

-Matt
Matthew Dillon

<dillon@backplane.com>

#3 - 05/11/2007 09:09 AM - corecode

bump date! :)

cheers
simon

ps: bitfields are natural beauties, macros need to wear makeup

#4 - 05/21/2007 11:43 PM - hasso

Does attached patch looks better?

#5 - 05/22/2007 12:44 AM - dillon

```
:+  data_len = ip_len - ((u_char *)cmn_hdr - (u_char *)ip);
:+
:...
:+  buf += sizeof(*cmn_hdr);
:+  data_len -= sizeof(*cmn_hdr);
:+
:+  while (data_len > 0) {
:+  obj_hdr = (struct icmp_ext_obj_hdr *)buf;
:+  obj_len = ntohs(obj_hdr->length);
```

This should be data_len >= sizeof(struct icmp_ext_obj_hdr),
not >= 0.

```
while (data_len >= sizeof(struct icmp_ext_obj_hdr)) {
...

:+
:+  /*
:+   * Sanity check the length field
:+   */
:+  if (obj_len > data_len) {
:+   return;
:+  }
```

obj_len can be 0. Check that obj_len < sizeof(*obj_hdr)
and return if it isn't. obj_len can be an odd number, which
is also not a good idea. Note sure about alignment requirements,
it might even have to be 4-byte aligned. But 2-byte for sure.

```
if (obj_len < sizeof(*obj_hdr) || obj_len > data_len)
return;
if (obj_len & 3) /* either & 1 or & 3, depending */
return;

:+
:+  data_len -= obj_len;
:+
:+  /*
:+   * Move past the object header
:+   */
:+  buf += sizeof(struct icmp_ext_obj_hdr);
:+  obj_len -= sizeof(struct icmp_ext_obj_hdr);
:+
:+  switch (obj_hdr->class_num) {
:+  case MPLS_STACK_ENTRY_CLASS:
:+   switch (obj_hdr->c_type) {
:+   case MPLS_STACK_ENTRY_C_TYPE:
:+    while (obj_len >= (int)sizeof(uint32_t)) {
:+     mpls_hdr = ntohl(*(uint32_t *)buf);
:+
:+     buf += sizeof(uint32_t);
:+     obj_len -= sizeof(uint32_t);
```

```
:+     printf(" [MPLS: Label %d Exp %d]",
:+         MPLS_LABEL(mpls_hdr), MPLS_EXP(mpls_hdr));
:+    }
:+    if (obj_len > 0) {
:+     /*
:+      * Something went wrong, and we're at
:+      * a unknown offset into the packet,
:+      * ditch the rest of it.
:+      */
:+     return;
:+    }
:+    break;
:+   default:
:+    /*
:+     * Unknown object, skip past it
:+     */
:+    buf += ntohs(obj_hdr->length) -
:+        sizeof(struct icmp_ext_obj_hdr);
```

Hmm. This is a bit messy

```
:...
:+   /*
:+    * Unknown object, skip past it
:+    */
:+   buf += ntohs(obj_hdr->length) -
:+       sizeof(struct icmp_ext_obj_hdr);
```

Same... a bit messy.

I suggest declaring a 'nextbuf' which you calculate up at the
'Move past the object header' code and just assign buf = nextbuf
in these two places.

```
:+   break;
:+  }
:+ }
: }
```

-Matt
Matthew Dillon
<dillon@backplane.com>

#6 - 05/22/2007 08:43 AM - hasso

Agreed.

RFC4884 says that "Each extension object contains one or more 32-bit words."
So, no question here IMHO.

Done. Updated patch is attached. Also fixed pointers to the standards
in progress.

#7 - 05/22/2007 04:58 PM - dillon

:Done. Updated patch is attached. Also fixed pointers to the standards
:in progress.
:
:--
:Hasso Tepper

Cool, looks good! Any MPLS enabled sites out there we can test it
with? In anycase, I say commit it.

-Matt

#8 - 05/22/2007 05:18 PM - nant

I'll setup an mpls lab scenario at work if i find some time and test it.

Cheers,
Nuno

#9 - 05/23/2007 05:46 AM - hasso

Many (if not most) big ISP's are using MPLS in their backbones. But it's
not always visible to others - routers don't have support for MPLS
extensions etc. For me traces to the www.kame.net and www.cogentco.com
have been testcases (besides the uses in my own network).

#10 - 05/23/2007 08:16 AM - hasso

Committed.

#11 - 05/23/2007 04:40 PM - dillon

:On Tuesday 22 May 2007 Matthew Dillon wrote:
:>   Cool, looks good! Any MPLS enabled sites out there we can test it
:>   with?
:
:Many (if not most) big ISP's are using MPLS in their backbones. But it's
:not always visible to others - routers don't have support for MPLS
:extensions etc. For me traces to the www.kame.net and www.cogentco.com
:have been testcases (besides the uses in my own network).
:

:--
:Hasso Tepper

Yahhh! I actually got MPLS output to display from cogentco.

-Matt
Matthew Dillon
<dillon@backplane.com>

#12 - 05/23/2007 10:16 PM - nant

As promised:

```
whitewidow# ./traceroute -M 192.168.255.4
traceroute to 192.168.255.4 (192.168.255.4), 64 hops max, 40 byte packets
1  10.29.134.154 (10.29.134.154)  1.25 ms  0.698 ms  0.659 ms
2  192.168.0.2 (192.168.0.2)  4.771 ms  4.749 ms  4.604 ms
3  192.168.255.2 (192.168.255.2)  6.811 ms  6.690 ms  6.782 ms [MPLS:
Label 16 Exp 0]
4  192.168.2.2 (192.168.2.2)  7.248 ms  6.753 ms  6.679 ms
5  192.168.255.4 (192.168.255.4)  7.708 ms  7.631 ms  7.622 ms
whitewidow#
```

And on the router:

```
Budapest:r1#sh mpls tunnels brief
name/id              destination        metric    state/label/intf
to-r3              192.168.255.3      A1        Out 16
on ATM2/0.1033.mpls
to-r3:r1:Budapest
lsp-c0a8ff03-1-35     192.168.255.1      R0        In  implicit-null
on ATM2/0.1033.mpls
to-r1:r3:Budapest
Budapest:r1#
```

Neat! Thanks,
Nuno

Files

icmp-mpls.patch	6.51 KB	05/22/2007	hasso
icmp-mpls.patch	6.53 KB	05/22/2007	hasso