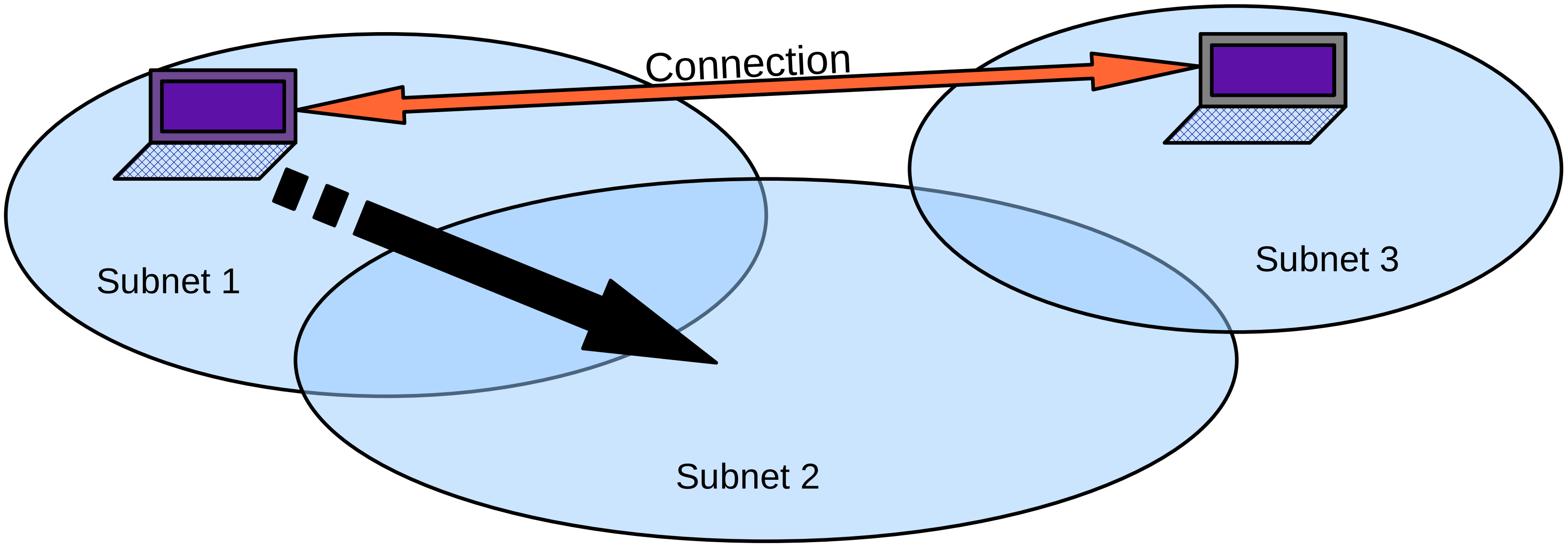
Socket (Session) Aware Change of IP - SACIP network functionality

Samo Pogačnik

Key notes about SACIP

- On-the-fly changes of network access point of a (mobile) user / endpoint device
- Possibility for preserving established network connections
- Application independency?



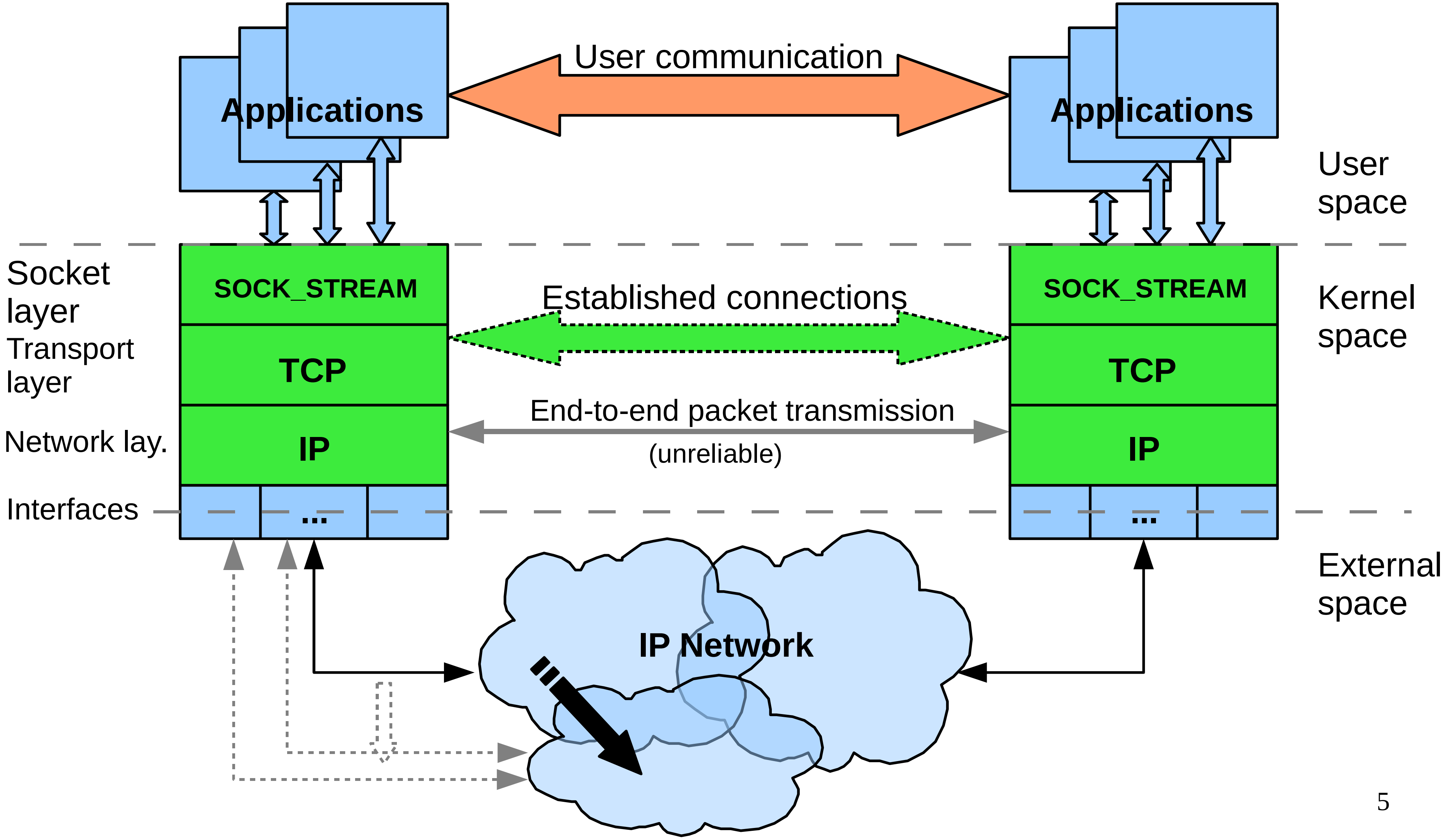
Motivation

- Mobile devices and wireless networks:
  - Multiple interfaces (access technologies)
  - Local areas covered by wireless IP networks
  - Areas covered by multiple IP networks:
    - borders of local areas
    - multiple access technologies
    - multiple providers
- True mobility:
  - Smooth and unnoticed switching between available access technologies, providers and local areas
  - Network access point (IP) changes

General idea

- Two facts:
  - IP layer delivers packets through a network independently of the upper (application) layers.
  - Network access point (IP address, local routing) change by itself does not prevent transmission and reception of packets (if packets contain correct values).
- To preseve existing connections:
  - Remote sides must be informed about the IP address change.
  - Application layers have to be adapted to the new IP address (very application specific).

Connected sockets



Functionality limitations

- Ignoring security and reliability issues
- No connection transfer to another network interface of a device
- Just simple network configuration (no NAT in the connection path)
- Ipv4 only
- Not possible to preserve connection, when old IP conectivity already lost
- Only TCP connected sockets tested (telnet)

Minimal scenario

- The simplest change of the network access point represents an IP change within the same subnet.
- New IP gets assigned as the secondary IP of the same interface and no route reconfiguration needed.
- The promote secondaries kernel option must be enabled.
- On deletion of the primary IP address (via ip tool):
  - SACIP functionality is called
  - Secondary IP becomes primary

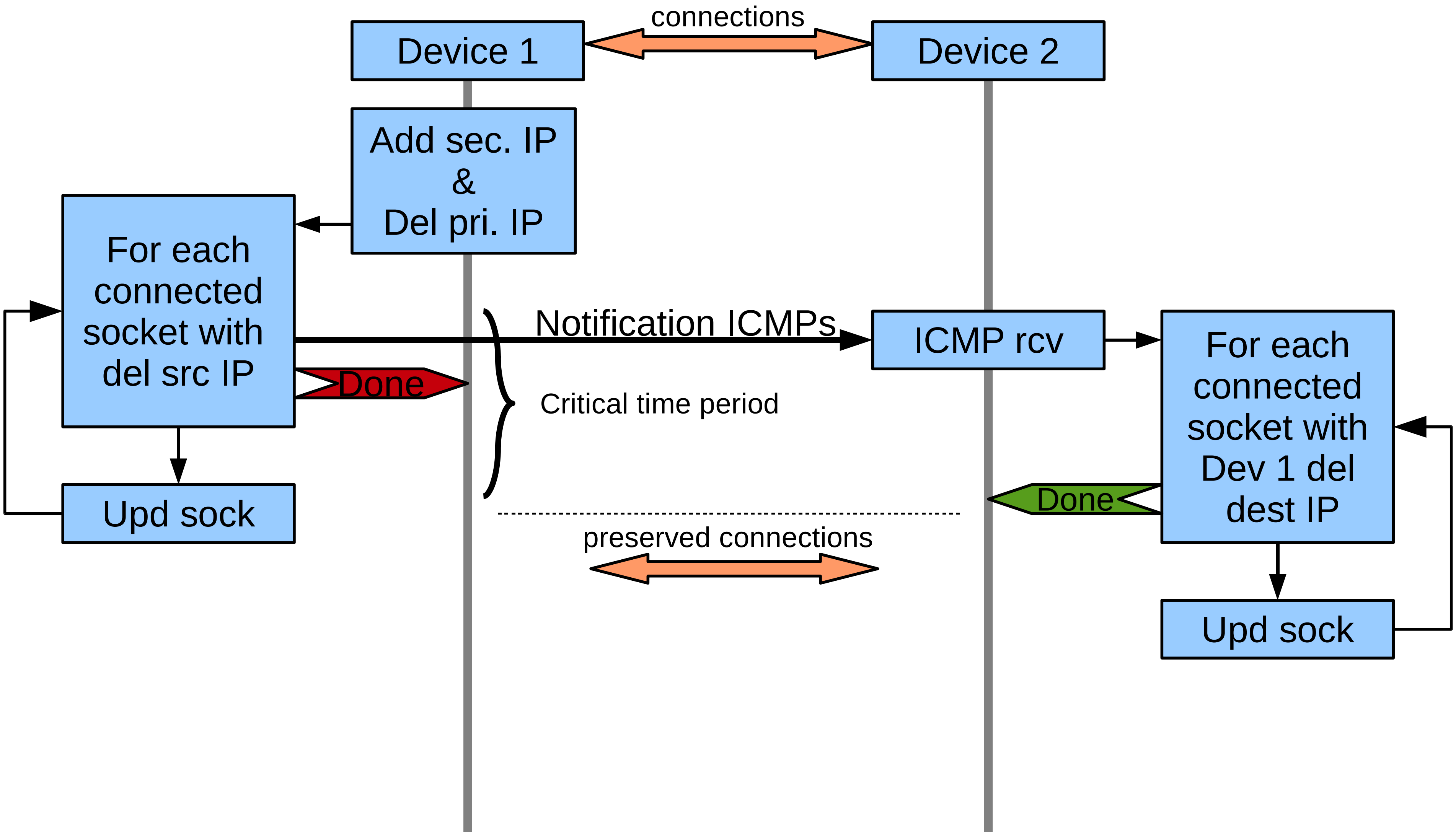
Scenario – local

- When SACIP gets called on the local side:
  - Connected sockets using changed IP addres are being searched for
  - For each connected socket found:
    - A notification (modified ICMP) message is sent to the connected party. This message's source address is still an old one and the message payload contains new IP address value.
    - Socket parameters are being updated with a new value (own addresses).
  - Now deletion of primary IP address finishes and packets of existing connections use new source IP address.

Scenario – remote

- On a receipt of the notification message on the remote side, remote SACIP functionality is called:
  - Similary, connected sockets using changed remote address are being searched for and socket parameters updated (partner addresses).
  - Afterwards outgoing packets of existing connections already use new destination IP address.

Scenario in picture



Implemenation

- To be able to perform these actions, socket structure has been extended:
  - added two additional pairs of IP addresses (source and destination pair) to the inet socket structure
  - added index for the currently active IP address of each new pair
- The role of the original socket parameters has been split between the original and new parameters.

Implementation – cont.

- Socket structure initialization
- Replacements of original socket parameters:
  - Socket match for every packet received, ...
- Local SACIP activation on IP deletion:
  - Search for affected socket, send notification, update socket params
- ICMP notification message
- Remote SACIP activation on the ICMP notification receipt

The socket structure

- Inet socket extension:

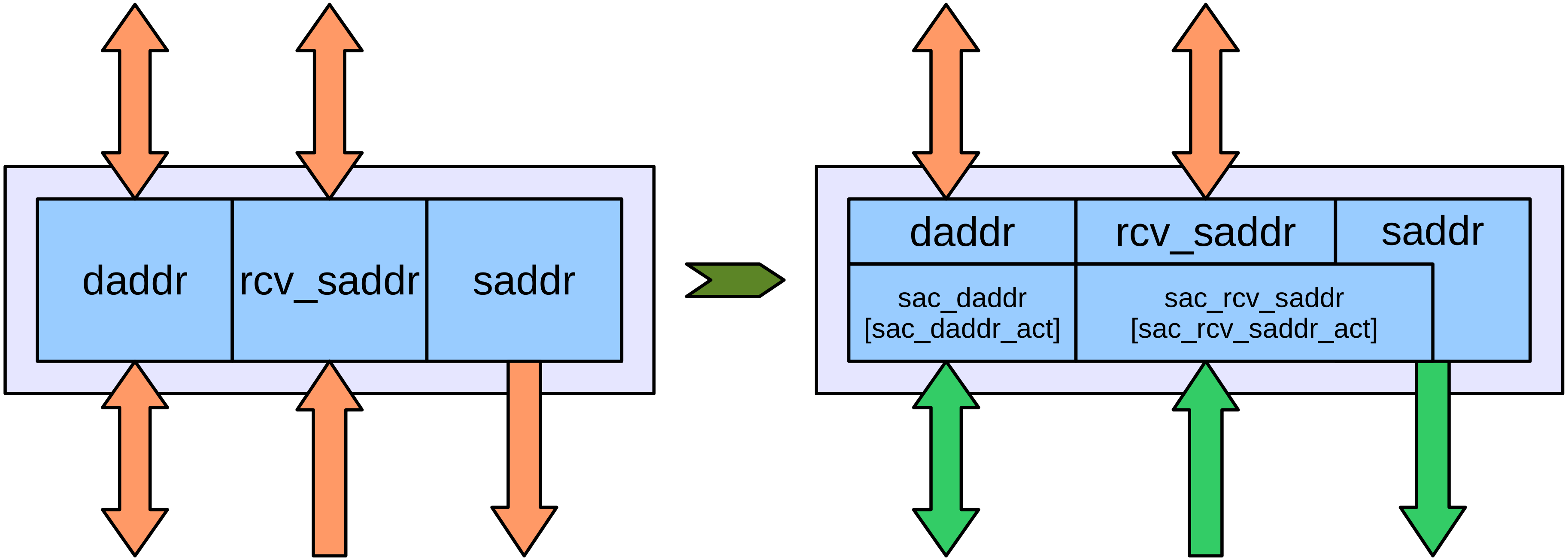
```
diff -Nurp linux-2.6.19/include/net/inet_sock.h linux-2.6.19-sacip/include/net/inet_sock.h
--- linux-2.6.19/include/net/inet_sock.h  2007-01-04 22:40:25.000000000 +0100
+++ linux-2.6.19-sacip/include/net/inet_sock.h 2007-09-13 22:56:17.000000000 +0200
@@ -112,6 +112,12 @@ struct inet_sock {
        /* Socket demultiplex comparisons on incoming packets. */
        __be32                 daddr;
        __be32                 rcv_saddr;
+#ifdef CONFIG_SACIP
+       __be32                 sac_daddr[2];
+       int               sac_daddr_act;
+       __be32                 sac_rcv_saddr[2];
+       int               sac_rcv_saddr_act;
+#endif
        __be16                 dport;
        __u16                  num;
        __be32                 saddr;
```

- Helper functions for the extension manipulation:

```
sac_inet_rcv_saddr(),  sac_init_rcv_saddr(),   sac_add_rcv_saddr(),    sac_act_rcv_saddr()
sac_inet_daddr(),      sac_init_daddr(),       sac_add_daddr(),        sac_act_daddr()
```

Socket parameter roles

Application socket interaction



Transport and Network socket interaction

Socket match

```
#ifndef CONFIG_SACIP
#define INET_MATCH(__sk, __hash, __cookie, __saddr, __daddr, __ports, __dif)    \
        (((__sk)->sk_hash == (__hash))                          &&       \
         (inet_sk(__sk)->daddr          == (__saddr))           &&       \
         (inet_sk(__sk)->rcv_saddr      == (__daddr))           &&       \
         ((*((__portpair *)&(inet_sk(__sk)->dport))) == (__ports))      &&       \
         (!((__sk)->sk_bound_dev_if) || ((__sk)->sk_bound_dev_if == (__dif))))
#define INET_TW_MATCH(__sk, __hash,__cookie, __saddr, __daddr, __ports, __dif)  \
        (((__sk)->sk_hash == (__hash))                          &&       \
         (inet_twsk(__sk)->tw_daddr     == (__saddr))           &&       \
         (inet_twsk(__sk)->tw_rcv_saddr == (__daddr))           &&       \
         ((*((__portpair *)&(inet_twsk(__sk)->tw_dport))) == (__ports)) &&       \
         (!((__sk)->sk_bound_dev_if) || ((__sk)->sk_bound_dev_if == (__dif))))
#else
#define INET_MATCH(__sk, __hash, __cookie, __saddr, __daddr, __ports, __dif) \
        (((__sk)->sk_hash == (__hash)) && \
         (sac_inet_daddr(__sk) == (__saddr)) && \
         (sac_inet_rcv_saddr(__sk) == (__daddr)) && \
         ((*((__portpair *)&(inet_sk(__sk)->dport))) == (__ports)) && \
         (!((__sk)->sk_bound_dev_if) || ((__sk)->sk_bound_dev_if == (__dif))))
#define INET_TW_MATCH(__sk, __hash,__cookie, __saddr, __daddr, __ports, __dif) \
        (((__sk)->sk_hash == (__hash)) && \
         (sac_inet_tw_daddr(__sk) == (__saddr)) && \
         (sac_inet_tw_rcv_saddr(__sk) == (__saddr)) && \
         ((*((__portpair *)&(inet_twsk(__sk)->tw_dport))) == (__ports)) && \
         (!((__sk)->sk_bound_dev_if) || ((__sk)->sk_bound_dev_if == (__dif))))
#endif
```

Local activation

```
void sac_add_rcv_saddr_tcp(__be32 orig, __be32 new)
{
        int bucket = 0;

        for (bucket = 0; bucket < tcp_hashinfo.ehash_size; ++bucket) {
                struct sock *sk;
                struct hlist_node *node;

                read_lock(&tcp_hashinfo.ehash[bucket].lock);
                sk_for_each(sk, node, &tcp_hashinfo.ehash[bucket].chain) {
                        if (sk->sk_family != AF_INET) {
                                continue;
                        }
                        if (sac_inet_rcv_saddr(sk) == orig) {
                                icmp_sacip_send(sk, ICMP_SACIP, 0, new);
                                read_unlock(&tcp_hashinfo.ehash[bucket].lock);
                                inet_unhash(&tcp_hashinfo, sk);
                                sac_add_rcv_saddr(inet_sk(sk), new);
                                sac_act_rcv_saddr(inet_sk(sk));
                                inet_sk(sk)->saddr = new;
                                inet_hash(&tcp_hashinfo, sk);
                                read_lock(&tcp_hashinfo.ehash[bucket].lock);
                        }
                }
                read_unlock(&tcp_hashinfo.ehash[bucket].lock);
        }

}
```

Notification ICMP

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |     Code      |          Checksum             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 New IP Address of sending device              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
diff -Nurp linux-2.6.19/include/linux/icmp.h linux-2.6.19-sacip/include/linux/icmp.h
--- linux-2.6.19/include/linux/icmp.h   2007-01-04 22:40:25.000000000 +0100
+++ linux-2.6.19-sacip/include/linux/icmp.h  2007-09-13 22:56:17.000000000 +0200
@@ -32,7 +32,12 @@
 #define ICMP_INFO_REPLY      16    /* Information Reply         */
 #define ICMP_ADDRESS         17    /* Address Mask Request      */
 #define ICMP_ADDRESSREPLY    18    /* Address Mask Reply        */
+#ifndef CONFIG_SACIP
 #define NR_ICMP_TYPES        18
+#else
+#define ICMP_SACIP           20    /* Session Aware Change of IP */
+#define NR_ICMP_TYPES        20
+#endif
```

- ICMP type 20 as specified by IANA:
  - 20-29 Reserved (for Robustness Experiment)   [ZSu]

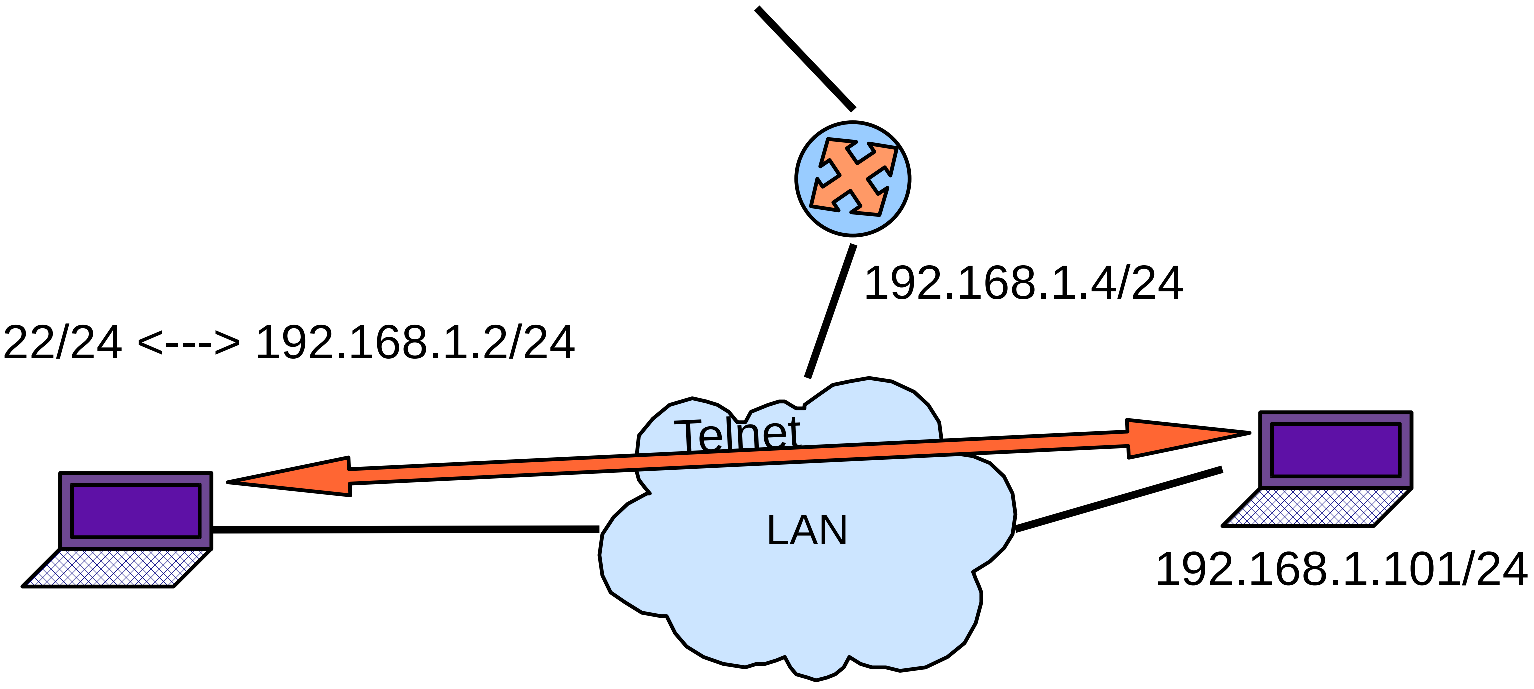
Remote activation

```
void sac_add_daddr_tcp(__be32 orig, __be32 new)
{
        int bucket = 0;

        for (bucket = 0; bucket < tcp_hashinfo.ehash_size; ++bucket) {
                struct sock *sk;
                struct hlist_node *node;

                read_lock(&tcp_hashinfo.ehash[bucket].lock);
                sk_for_each(sk, node, &tcp_hashinfo.ehash[bucket].chain) {
                        if (sk->sk_family != AF_INET) {
                                continue;
                        }
                        if (sac_inet_daddr(sk) == orig) {
                                read_unlock(&tcp_hashinfo.ehash[bucket].lock);
                                inet_unhash(&tcp_hashinfo, sk);
                                sac_add_daddr(inet_sk(sk), new);
                                sac_act_daddr(inet_sk(sk));
                                inet_hash(&tcp_hashinfo, sk);
                                read_lock(&tcp_hashinfo.ehash[bucket].lock);
                                sk_dst_reset(sk);
                        }
                }
                read_unlock(&tcp_hashinfo.ehash[bucket].lock);
        }

}
```

Test examples

1) IP change (the same subnet)

2) IP change (from one subnet to another in the same broadcast domain - default router involved)



1) 192.168.1.22/24 <---> 192.168.1.2/24

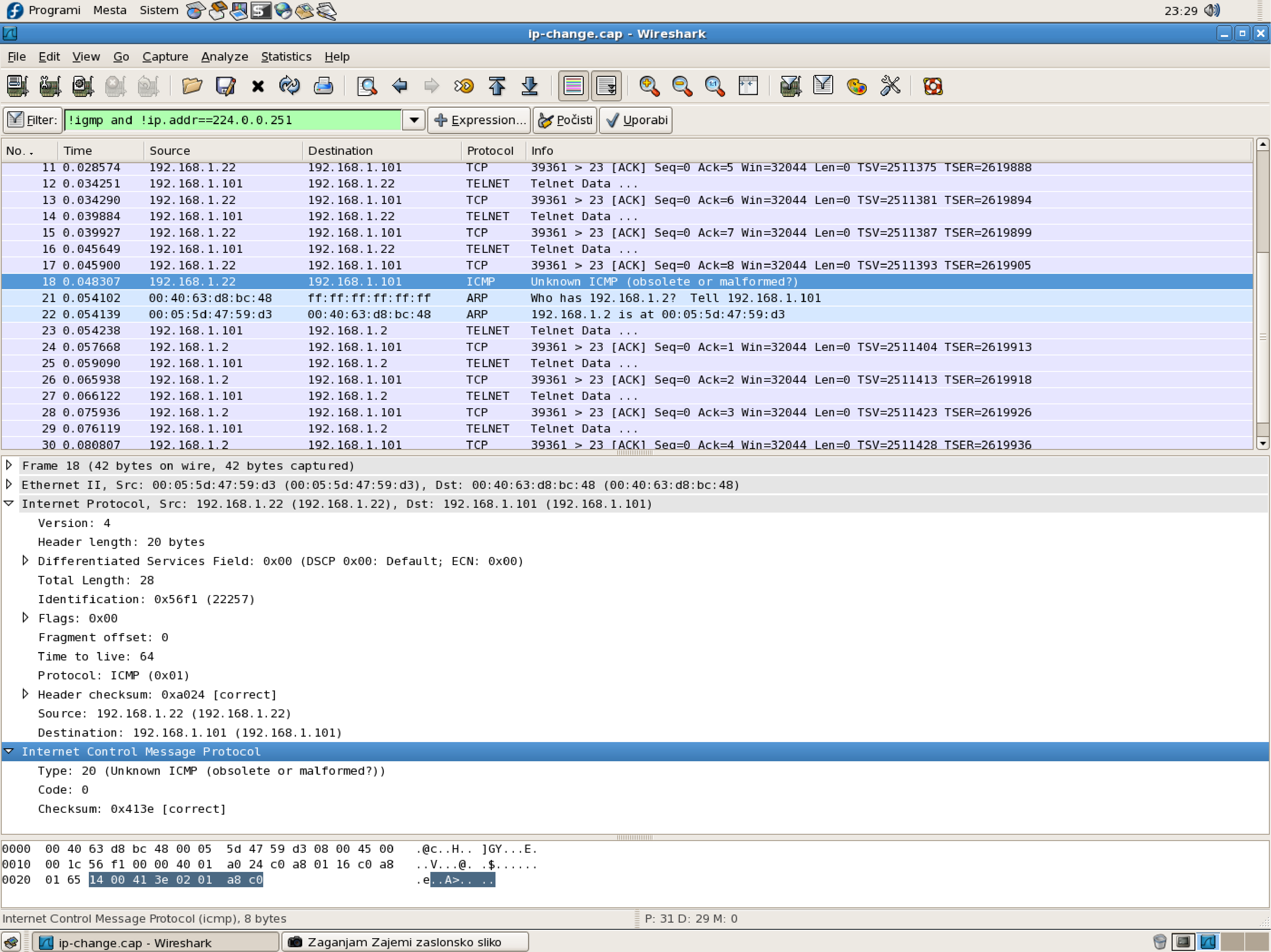
2) 192.168.1.2/25 <---> 192.168.1.222/25

IP change

```
[root@localhost samo]#  /sbin/ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:05:5d:47:59:d3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.22/24 scope global eth0
    inet6 fe80::205:5dff:fe47:59d3/64 scope link
       valid_lft forever preferred_lft forever
[root@localhost samo]#  /sbin/ip addr add 192.168.1.2/24 dev eth0
[root@localhost samo]#  /sbin/ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:05:5d:47:59:d3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.22/24 scope global eth0
    inet 192.168.1.2/24 scope global secondary eth0
    inet6 fe80::205:5dff:fe47:59d3/64 scope link
       valid_lft forever preferred_lft forever
[root@localhost samo]# /sbin/ip addr del 192.168.1.22/24 dev eth0
[root@localhost samo]#  /sbin/ip addr show dev eth 0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:05:5d:47:59:d3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.2/24 scope global eth0
    inet6 fe80::205:5dff:fe47:59d3/64 scope link
       valid_lft forever preferred_lft forever
```

IP change – cont. 1

```
[root@localhost samo]# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 0.0.0.0:139                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:23                  0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:602                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:445                 0.0.0.0:*                   LISTEN
tcp        0      0 192.168.1.22:46915          192.168.1.101:23            ESTABLISHED
tcp        0      0 :::3690                     :::*                        LISTEN
tcp        0      0 :::22                       :::*                        LISTEN
[root@localhost samo]# netstat -nat
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address               Foreign Address             State
tcp        0      0 0.0.0.0:139                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:111                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:23                  0.0.0.0:*                   LISTEN
tcp        0      0 127.0.0.1:631               0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:602                 0.0.0.0:*                   LISTEN
tcp        0      0 0.0.0.0:445                 0.0.0.0:*                   LISTEN
tcp        0      0 192.168.1.2:46915           192.168.1.101:23            ESTABLISHED
tcp        0      0 :::3690                     :::*                        LISTEN
tcp        0      0 :::22                       :::*                        LISTEN
```



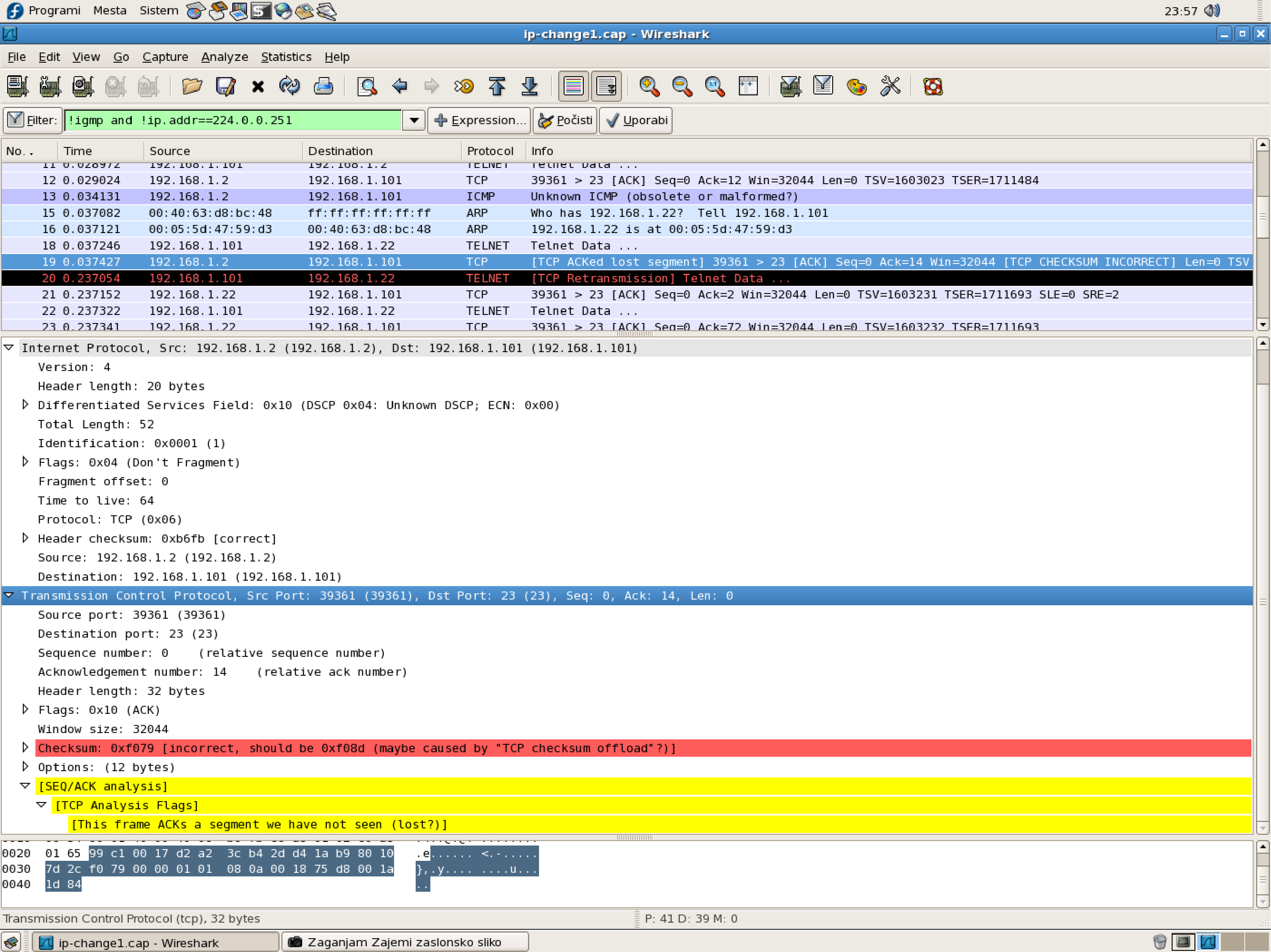
Filter: !igmp and !ip.addr==224.0.0.251

```
No. .  Time      Source             Destination        Protocol  Info
   11 0.028574  192.168.1.22       192.168.1.101      TCP       39361 > 23 [ACK] Seq=0 Ack=5 Win=32044 Len=0 TSV=2511375 TSER=2619888
   12 0.034251  192.168.1.101      192.168.1.22       TELNET    Telnet Data ...
   13 0.034290  192.168.1.22       192.168.1.101      TCP       39361 > 23 [ACK] Seq=0 Ack=6 Win=32044 Len=0 TSV=2511381 TSER=2619894
   14 0.039884  192.168.1.101      192.168.1.22       TELNET    Telnet Data ...
   15 0.039927  192.168.1.22       192.168.1.101      TCP       39361 > 23 [ACK] Seq=0 Ack=7 Win=32044 Len=0 TSV=2511387 TSER=2619899
   16 0.045649  192.168.1.101      192.168.1.22       TELNET    Telnet Data ...
   17 0.045900  192.168.1.22       192.168.1.101      TCP       39361 > 23 [ACK] Seq=0 Ack=8 Win=32044 Len=0 TSV=2511393 TSER=2619905
   18 0.048307  192.168.1.22       192.168.1.101      ICMP      Unknown ICMP (obsolete or malformed?)
   21 0.054102  00:40:63:d8:bc:48  ff:ff:ff:ff:ff:ff  ARP       Who has 192.168.1.2?  Tell 192.168.1.101
   22 0.054139  00:05:5d:47:59:d3  00:40:63:d8:bc:48  ARP       192.168.1.2 is at 00:05:5d:47:59:d3
   23 0.054238  192.168.1.101      192.168.1.2        TELNET    Telnet Data ...
   24 0.057668  192.168.1.2        192.168.1.101      TCP       39361 > 23 [ACK] Seq=0 Ack=1 Win=32044 Len=0 TSV=2511404 TSER=2619913
   25 0.059090  192.168.1.101      192.168.1.2        TELNET    Telnet Data ...
   26 0.065938  192.168.1.2        192.168.1.101      TCP       39361 > 23 [ACK] Seq=0 Ack=2 Win=32044 Len=0 TSV=2511413 TSER=2619918
   27 0.066122  192.168.1.101      192.168.1.2        TELNET    Telnet Data ...
   28 0.075936  192.168.1.2        192.168.1.101      TCP       39361 > 23 [ACK] Seq=0 Ack=3 Win=32044 Len=0 TSV=2511423 TSER=2619926
   29 0.076119  192.168.1.101      192.168.1.2        TELNET    Telnet Data ...
   30 0.080807  192.168.1.2        192.168.1.101      TCP       39361 > 23 [ACK] Seq=0 Ack=4 Win=32044 Len=0 TSV=2511428 TSER=2619936
```

```
▷ Frame 18 (42 bytes on wire, 42 bytes captured)
▷ Ethernet II, Src: 00:05:5d:47:59:d3 (00:05:5d:47:59:d3), Dst: 00:40:63:d8:bc:48 (00:40:63:d8:bc:48)
▽ Internet Protocol, Src: 192.168.1.22 (192.168.1.22), Dst: 192.168.1.101 (192.168.1.101)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 28
    Identification: 0x56f1 (22257)
  ▷ Flags: 0x00
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (0x01)
  ▷ Header checksum: 0xa024 [correct]
    Source: 192.168.1.22 (192.168.1.22)
    Destination: 192.168.1.101 (192.168.1.101)
▽ Internet Control Message Protocol
    Type: 20 (Unknown ICMP (obsolete or malformed?))
    Code: 0
    Checksum: 0x413e [correct]
```

```
0000  00 40 63 d8 bc 48 00 05  5d 47 59 d3 08 00 45 00   .@c..H.. ]GY...E.
0010  00 1c 56 f1 00 00 40 01  a0 24 c0 a8 01 16 c0 a8   ..V...@. .$......
0020  01 65 14 00 41 3e 02 01  a8 c0                     .e..A>.. ..
```

Internet Control Message Protocol (icmp), 8 bytes   P: 31 D: 29 M: 0



Filter: !igmp and !ip.addr==224.0.0.251

```
No. .  Time      Source              Destination         Protocol  Info
 11 0.028972   192.168.1.2         192.168.1.2         TELNET    Telnet Data ...
 12 0.029024   192.168.1.2         192.168.1.101       TCP       39361 > 23 [ACK] Seq=0 Ack=12 Win=32044 Len=0 TSV=1603023 TSER=1711484
 13 0.034131   192.168.1.2         192.168.1.101       ICMP      Unknown ICMP (obsolete or malformed?)
 15 0.037082   00:40:63:d8:bc:48   ff:ff:ff:ff:ff:ff   ARP       Who has 192.168.1.22?  Tell 192.168.1.101
 16 0.037121   00:05:5d:47:59:d3   00:40:63:d8:bc:48   ARP       192.168.1.22 is at 00:05:5d:47:59:d3
 18 0.037246   192.168.1.101       192.168.1.22        TELNET    Telnet Data ...
 19 0.037427   192.168.1.2         192.168.1.101       TCP       [TCP ACKed lost segment] 39361 > 23 [ACK] Seq=0 Ack=14 Win=32044 [TCP CHECKSUM INCORRECT] Len=0 TSV
 20 0.237054   192.168.1.101       192.168.1.22        TELNET    [TCP Retransmission] Telnet Data ...
 21 0.237152   192.168.1.22        192.168.1.101       TCP       39361 > 23 [ACK] Seq=0 Ack=2 Win=32044 Len=0 TSV=1603231 TSER=1711693 SLE=0 SRE=2
 22 0.237322   192.168.1.101       192.168.1.22        TELNET    Telnet Data ...
 23 0.237341   192.168.1.22        192.168.1.101       TCP       39361 > 23 [ACK] Seq=0 Ack=72 Win=32044 Len=0 TSV=1603232 TSER=1711693
```

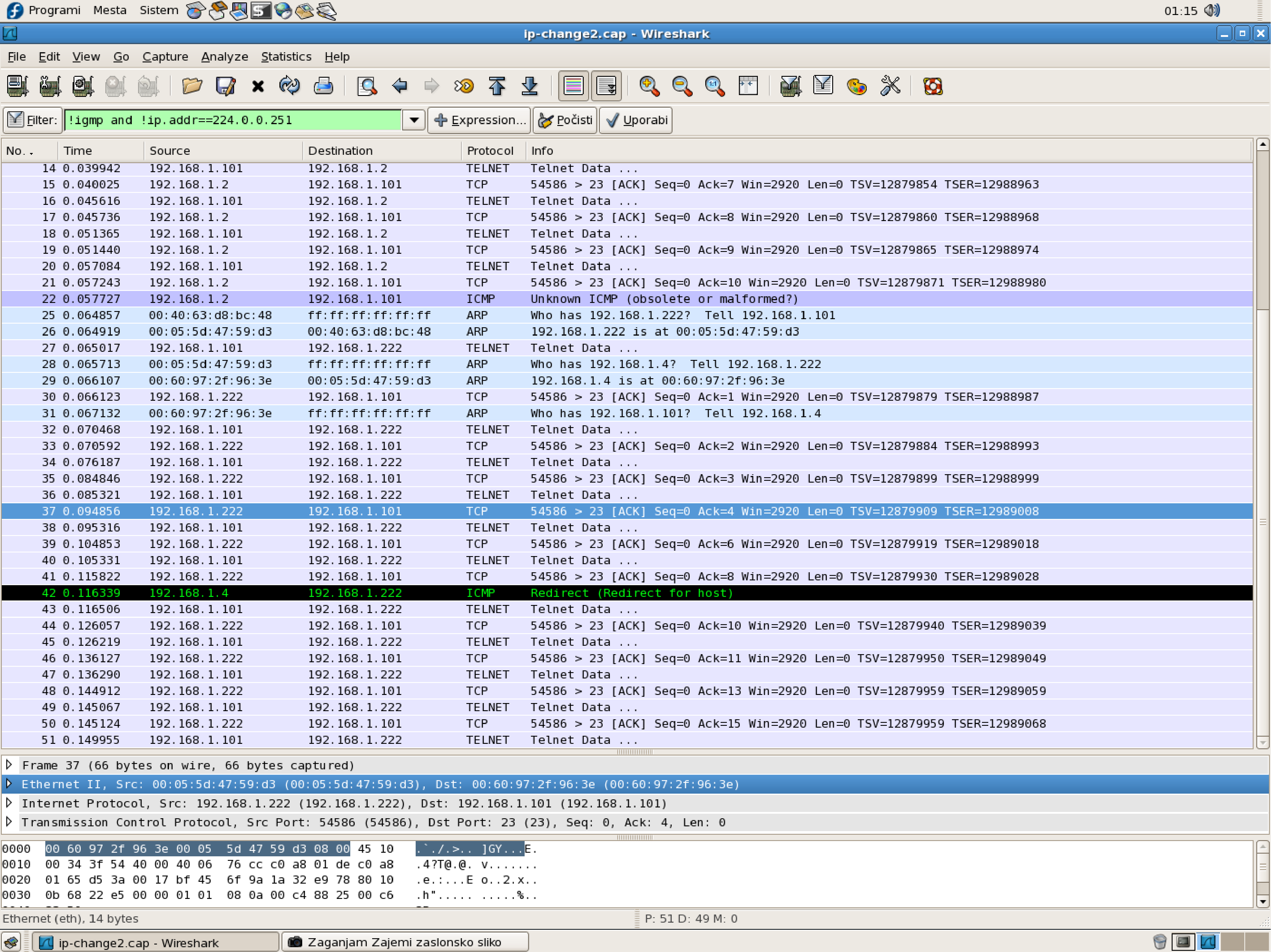
```
▽ Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.101 (192.168.1.101)
    Version: 4
    Header length: 20 bytes
  ▷ Differentiated Services Field: 0x10 (DSCP 0x04: Unknown DSCP; ECN: 0x00)
    Total Length: 52
    Identification: 0x0001 (1)
  ▷ Flags: 0x04 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (0x06)
  ▷ Header checksum: 0xb6fb [correct]
    Source: 192.168.1.2 (192.168.1.2)
    Destination: 192.168.1.101 (192.168.1.101)
▽ Transmission Control Protocol, Src Port: 39361 (39361), Dst Port: 23 (23), Seq: 0, Ack: 14, Len: 0
    Source port: 39361 (39361)
    Destination port: 23 (23)
    Sequence number: 0    (relative sequence number)
    Acknowledgement number: 14    (relative ack number)
    Header length: 32 bytes
  ▷ Flags: 0x10 (ACK)
    Window size: 32044
  ▷ Checksum: 0xf079 [incorrect, should be 0xf08d (maybe caused by "TCP checksum offload"?)]
  ▷ Options: (12 bytes)
  ▽ [SEQ/ACK analysis]
    ▽ [TCP Analysis Flags]
        [This frame ACKs a segment we have not seen (lost?)]
```

```
0020  01 65 99 c1 00 17 d2 a2  3c b4 2d d4 1a b9 80 10   .e...... <.-.....
0030  7d 2c f0 79 00 00 01 01  08 0a 00 18 75 d8 00 1a   },.y.... ....u...
0040  1d 84                                              ..
```

Transmission Control Protocol (tcp), 32 bytes   P: 41 D: 39 M: 0

IP change – cont. 2

```
[root@localhost samo]#  /sbin/ip route show
192.168.1.0/25 dev eth0  proto kernel  scope link  src 192.168.1.2
169.254.0.0/16 dev eth0  scope link
default via 192.168.1.4 dev eth0
[root@localhost samo]#  /sbin/ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:05:5d:47:59:d3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.2/25 brd 192.168.1.127 scope global eth0
    ...
[root@localhost samo]#  /sbin/ip addr add 192.168.1.222/25 dev eth0
[root@localhost samo]#  /sbin/ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:05:5d:47:59:d3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.2/25 brd 192.168.1.127 scope global eth0
    inet 192.168.1.222/25 scope global eth0
    ...
[root@localhost samo]#  /sbin/ip addr del 192.168.1.2/25 dev eth0
[root@localhost samo]#  /sbin/ip addr show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:05:5d:47:59:d3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.222/25 scope global eth0
    ...
[root@localhost samo]#  /sbin/ip route show
192.168.1.128/25 dev eth0  proto kernel  scope link  src 192.168.1.222
169.254.0.0/16 dev eth0  scope link
default via 192.168.1.4 dev eth0
```



Filter: !igmp and !ip.addr==224.0.0.251

```
No. .  Time      Source             Destination        Protocol  Info
    14 0.039942  192.168.1.101      192.168.1.2        TELNET    Telnet Data ...
    15 0.040025  192.168.1.2        192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=7 Win=2920 Len=0 TSV=12879854 TSER=12988963
    16 0.045616  192.168.1.101      192.168.1.2        TELNET    Telnet Data ...
    17 0.045736  192.168.1.2        192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=8 Win=2920 Len=0 TSV=12879860 TSER=12988968
    18 0.051365  192.168.1.101      192.168.1.2        TELNET    Telnet Data ...
    19 0.051440  192.168.1.2        192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=9 Win=2920 Len=0 TSV=12879865 TSER=12988974
    20 0.057084  192.168.1.101      192.168.1.2        TELNET    Telnet Data ...
    21 0.057243  192.168.1.2        192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=10 Win=2920 Len=0 TSV=12879871 TSER=12988980
    22 0.057727  192.168.1.2        192.168.1.101      ICMP      Unknown ICMP (obsolete or malformed?)
    25 0.064857  00:40:63:d8:bc:48  ff:ff:ff:ff:ff:ff  ARP       Who has 192.168.1.222?  Tell 192.168.1.101
    26 0.064919  00:05:5d:47:59:d3  00:40:63:d8:bc:48  ARP       192.168.1.222 is at 00:05:5d:47:59:d3
    27 0.065017  192.168.1.101      192.168.1.222      TELNET    Telnet Data ...
    28 0.065713  00:05:5d:47:59:d3  ff:ff:ff:ff:ff:ff  ARP       Who has 192.168.1.4?  Tell 192.168.1.222
    29 0.066107  00:60:97:2f:96:3e  00:05:5d:47:59:d3  ARP       192.168.1.4 is at 00:60:97:2f:96:3e
    30 0.066123  192.168.1.222      192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=1 Win=2920 Len=0 TSV=12879879 TSER=12988987
    31 0.067132  00:60:97:2f:96:3e  ff:ff:ff:ff:ff:ff  ARP       Who has 192.168.1.101?  Tell 192.168.1.4
    32 0.070468  192.168.1.101      192.168.1.222      TELNET    Telnet Data ...
    33 0.070592  192.168.1.222      192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=2 Win=2920 Len=0 TSV=12879884 TSER=12988993
    34 0.076187  192.168.1.101      192.168.1.222      TELNET    Telnet Data ...
    35 0.084846  192.168.1.222      192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=3 Win=2920 Len=0 TSV=12879899 TSER=12988999
    36 0.085321  192.168.1.101      192.168.1.222      TELNET    Telnet Data ...
    37 0.094856  192.168.1.222      192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=4 Win=2920 Len=0 TSV=12879909 TSER=12989008
    38 0.095316  192.168.1.101      192.168.1.222      TELNET    Telnet Data ...
    39 0.104853  192.168.1.222      192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=6 Win=2920 Len=0 TSV=12879919 TSER=12989018
    40 0.105331  192.168.1.101      192.168.1.222      TELNET    Telnet Data ...
    41 0.115822  192.168.1.222      192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=8 Win=2920 Len=0 TSV=12879930 TSER=12989028
    42 0.116339  192.168.1.4        192.168.1.222      ICMP      Redirect (Redirect for host)
    43 0.116506  192.168.1.101      192.168.1.222      TELNET    Telnet Data ...
    44 0.126057  192.168.1.222      192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=10 Win=2920 Len=0 TSV=12879940 TSER=12989039
    45 0.126219  192.168.1.101      192.168.1.222      TELNET    Telnet Data ...
    46 0.136127  192.168.1.222      192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=11 Win=2920 Len=0 TSV=12879950 TSER=12989049
    47 0.136290  192.168.1.101      192.168.1.222      TELNET    Telnet Data ...
    48 0.144912  192.168.1.222      192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=13 Win=2920 Len=0 TSV=12879959 TSER=12989059
    49 0.145067  192.168.1.101      192.168.1.222      TELNET    Telnet Data ...
    50 0.145124  192.168.1.222      192.168.1.101      TCP       54586 > 23 [ACK] Seq=0 Ack=15 Win=2920 Len=0 TSV=12879959 TSER=12989068
    51 0.149955  192.168.1.101      192.168.1.222      TELNET    Telnet Data ...
```

```
▷ Frame 37 (66 bytes on wire, 66 bytes captured)
▷ Ethernet II, Src: 00:05:5d:47:59:d3 (00:05:5d:47:59:d3), Dst: 00:60:97:2f:96:3e (00:60:97:2f:96:3e)
▷ Internet Protocol, Src: 192.168.1.222 (192.168.1.222), Dst: 192.168.1.101 (192.168.1.101)
▷ Transmission Control Protocol, Src Port: 54586 (54586), Dst Port: 23 (23), Seq: 0, Ack: 4, Len: 0
```

```
0000  00 60 97 2f 96 3e 00 05  5d 47 59 d3 08 00 45 10   .`./.>.. ]GY...E.
0010  00 34 3f 54 40 00 40 06  76 cc c0 a8 01 de c0 a8   .4?T@.@. v.......
0020  01 65 d5 3a 00 17 bf 45  6f 9a 1a 32 e9 78 80 10   .e.:...E o..2.x..
0030  0b 68 22 e5 00 00 01 01  08 0a 00 c4 88 25 00 c6   .h"..... .....%..
```

Ethernet (eth), 14 bytes   P: 51 D: 49 M: 0

Complete solution

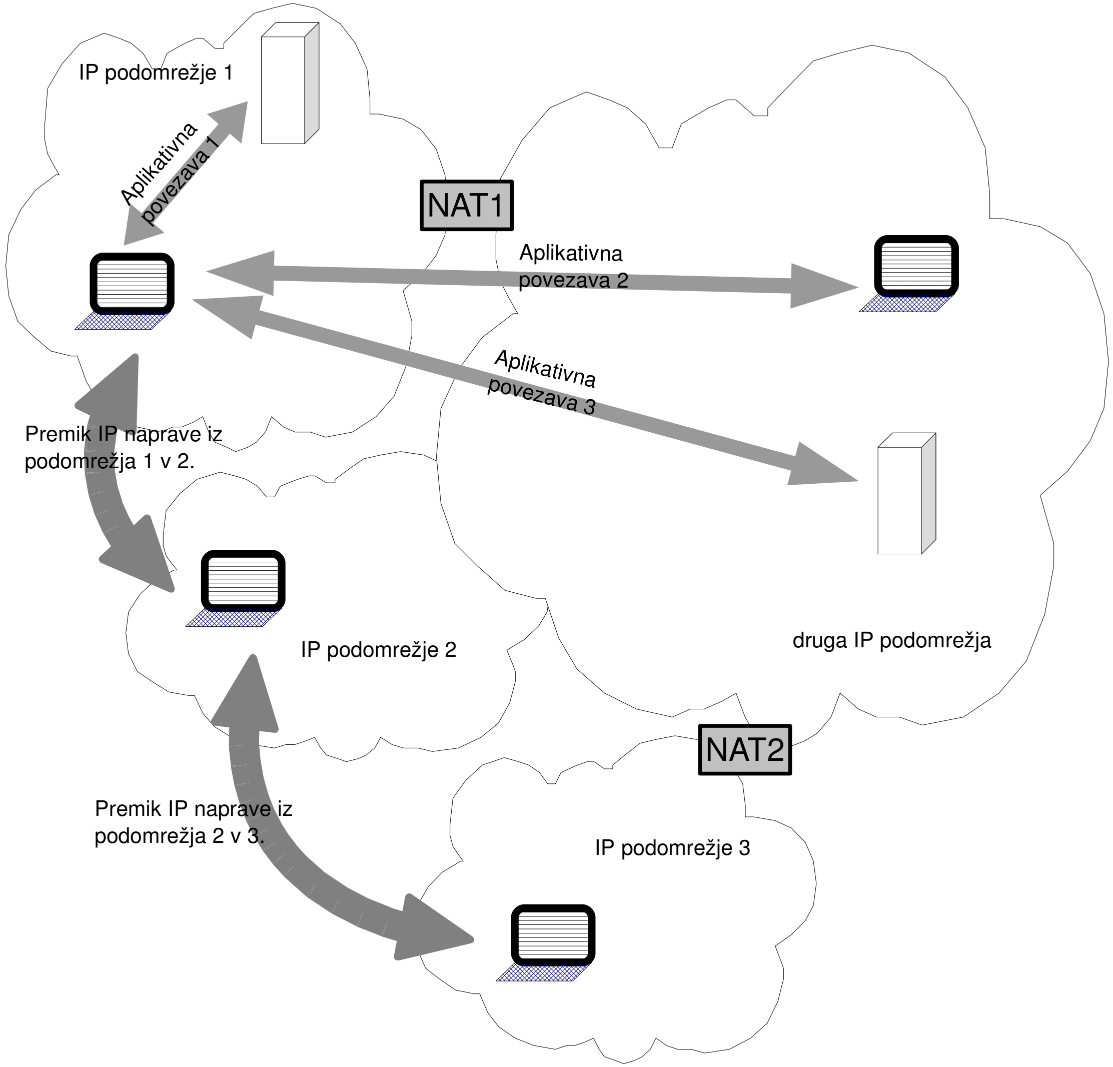
- Access point monitoring
- Security issues
- Temporary loss of the access point
- SACIP activation covering IP change, routing reconfiguration, interface change, ..., shared secret exchange, ...
- Application notification at both sides (requires application modifications) could resolve connection preservation for all types of connections.

Security

- It is very easy to send fake notification messages (man-in-the-middle attack)
- Encryption of notification messages and message format change:
  - Encrypted payload
  - Both old and new address in the payload
- Shared secrets; how to manage them (IPSec - SA, IKE; PKI - certificates, ...)

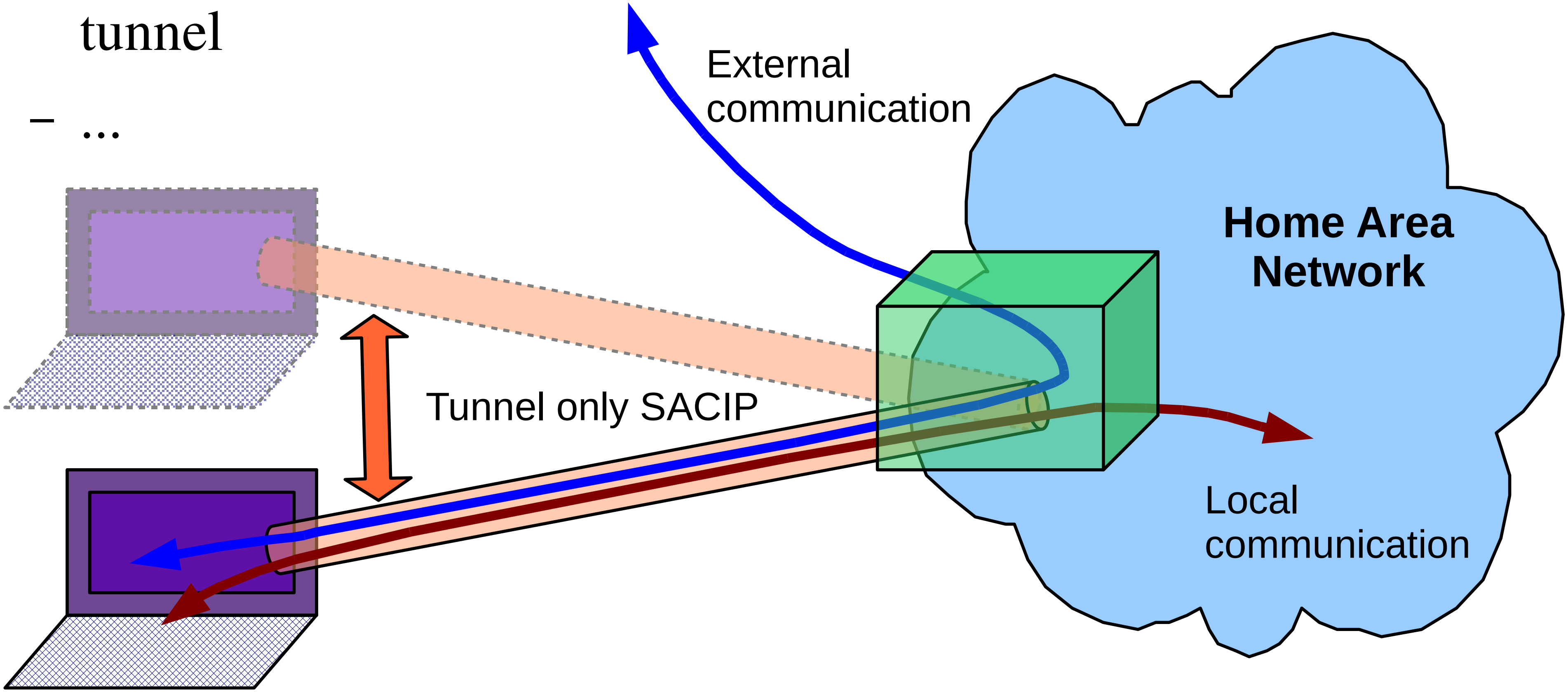
Possible enhancements?

- NAT in the PATH... Is it possible, needed, ...?



Possible enhancements...?

- A mobile tunnel (Mobile IP, VPN, IPSEC, ...)?
  - Any IP based tunnel could implement a SACIP like feature
  - Automatic preservation of all communication through the tunnel
  - ...



Thank you

- The WEB link to the SACIP patch:
  - http://84.255.254.67/patch-linux-2.6.19-sacip
  - some other things (old LTT++, ...)
- References:

[1] RFC 791, Internet Protocol, 1981

[2] RFC 793, Transmission Control Protocol, 1981

[3] RFC 768, User Datagram Protocol, 1980

[4] RFC 792, Internet Control Message Protocol, 1981

[5] RFC 854, Telnet Protocol, 1983

[6] Internet sockets, http://en.wikipedia.org/wiki/Internet_socket