Multi Event Protocol in the LHCb DAQ system: Linux implementation

Łukasz Olejnik

August 10, 2009

Plan

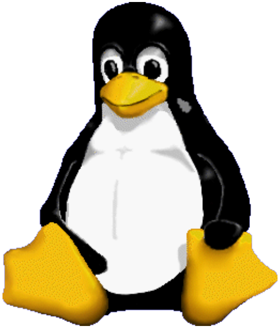
Outline

About me

Outline

Linux

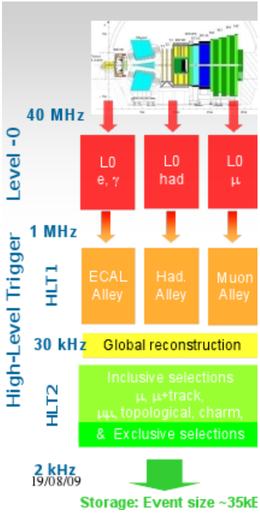
- Modern multi-user operating system
- Popular in server market, scientific and educational applications
- Also gaining popularity on user desktops (1% market share).



LHCb and its DAQ

A secret cavern located 100 m underground

- Tries to figure out what have happened with antimateria (CP Violation, rare decays, ...)
- For example, this is supposed to be a rare decay: $B_s \rightarrow \mu\mu$
- Basically a large room-T magnet and a few specialized detectors
- Triggers filter out the boring (?) stuff (as all other large-scale experiments do).
- (40 MHz) L0 - hardware, (1 MHz) HLT 1 & (30 kHz) 2 (2 kHz) - software based, using $O(2000)$ processors
- That's 35 $\mathrm{GB/s}$



Network layers

- Network people figured out that it's convenient to divide a big problem into smaller (but still huge) ones
- **OSI model**: (7) Application (6) Presentation (5) Session **(4) Transport (3) Network** (2) Data Link (1) Physical
- That's great, becouse it allows to create small teams which can focus on their specific work.
- Problem with this approach: a layer usually adds/removes an additional header or performs computations. Also, a lot of data copy takes place. This all results with overheads.

Figure: Communication between layers

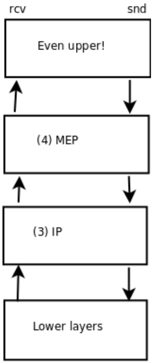
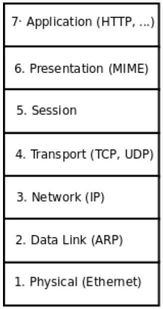
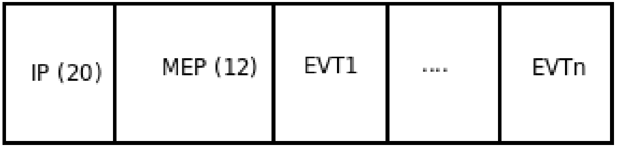


Figure: Network layers



Motivation
Intro to the protocol

- A custom, connection-less protocol. Have it's own headers and pseudoheaders
- A packet contains lots of event parts

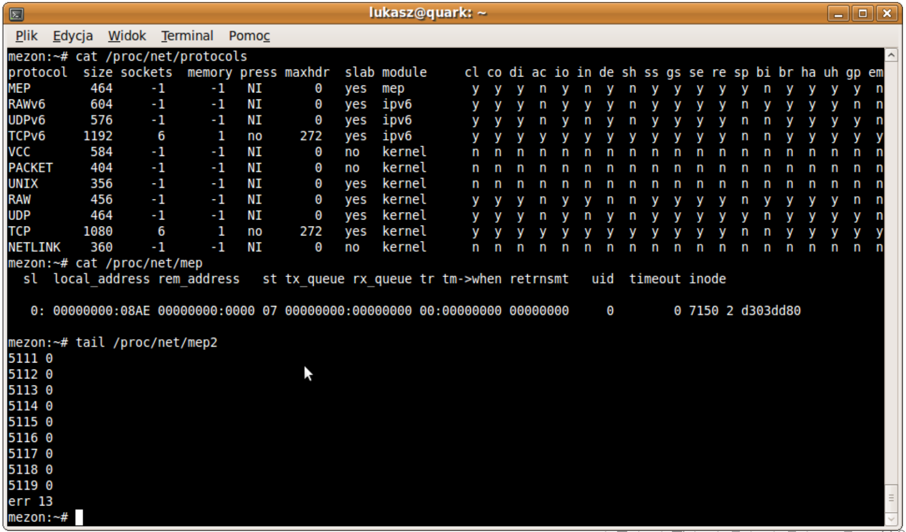


Changes?

- Currently: RAW sockets - packets come to the machine and then, Linux kernel directs them in a 'raw' form to the userland. So the user applications reads all packets (besides those handled by legitimate in-kernel protocols)
- **Linux implementation**: MEP works in L4
- But why? There is a power limit in The Pit (600 kW), so the more effective, the better (you can't just throw there a huge amounts of computers)
- Reduce the overheads wherever you can: leaves more resources for physics

Linux implementation

- Once a protocol is implemented ("introduced"), it can be elegantly used
- Both for testing and The Real thing. Theoretically should be even faster (there goes my Nobel prize)

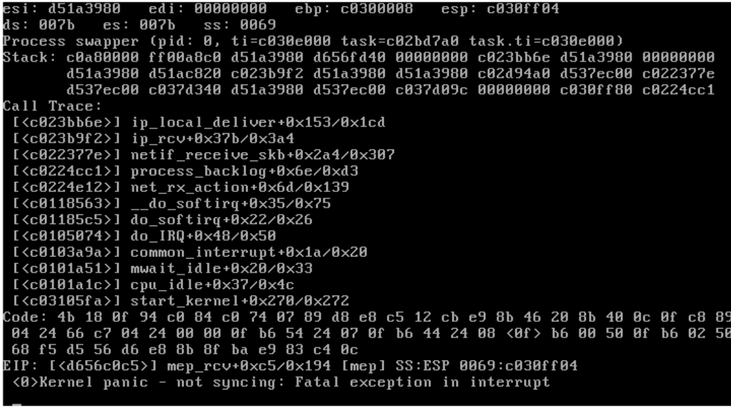


```
mezon:~# cat /proc/net/protocols
protocol  size sockets  memory press maxhdr  slab module     cl co di ac io in de sh ss gs se re sp bi br ha uh gp em
MEP        464      -1      -1   NI       0   yes  mep         y  y  y  n  y  n  y  n  y  y  y  y  y  n  y  y  y  y  n
RAWv6      604      -1      -1   NI       0   yes  ipv6        y  y  y  n  y  y  y  n  y  y  y  y  n  y  y  y  y  n  n
UDPv6      576      -1      -1   NI       0   yes  ipv6        y  y  y  n  y  n  y  n  y  y  y  y  n  n  y  y  y  y  n
TCPv6     1192       6       1   no     272   yes  ipv6        y  y  y  y  y  y  y  y  y  y  y  y  n  n  y  y  y  y  y
VCC        584      -1      -1   NI       0   no   kernel      n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n
PACKET     404      -1      -1   NI       0   no   kernel      n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n
UNIX       356      -1      -1   NI       0   yes  kernel      n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n
RAW        456      -1      -1   NI       0   yes  kernel      y  y  y  n  y  y  n  n  y  y  y  y  n  y  y  y  y  n  n
UDP        464      -1      -1   NI       0   yes  kernel      y  y  y  n  y  n  y  n  y  y  y  y  y  n  y  y  y  y  n
TCP       1080       6       1   no     272   yes  kernel      y  y  y  y  y  y  y  y  y  y  y  y  n  n  y  y  y  y  y
NETLINK    360      -1      -1   NI       0   no   kernel      n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n  n
mezon:~# cat /proc/net/mep
  sl  local_address rem_address   st tx_queue rx_queue tr tm->when retrnsmt   uid  timeout inode

   0: 00000000:08AE 00000000:0000 07 00000000:00000000 00:00000000 00000000     0        0 7150 2 d303dd80

mezon:~# tail /proc/net/mep2
5111 0
5112 0
5113 0
5114 0
5115 0
5116 0
5117 0
5118 0
5119 0
err 13
mezon:~# 
```

Summary

You can safely reset your computer now

- LHCb rocks the science!
- Linux rock the LHCb DAQ!
- I rock the MEP in the kernel.
  - Future: zero_copy from the network directly to the event builder (skip the kernel at all)
  - Someone else will rock this. Also, this will be **his worst nightmare**.
- But that was just 10 minutes...



```
esi: d51a3980   edi: 00000000   ebp: c0300008   esp: c030ff04
ds: 007b   es: 007b   ss: 0069
Process swapper (pid: 0, ti=c030e000 task=c02bd7a0 task.ti=c030e000)
Stack: c0a80000 ff00a8c0 d51a3980 d656fd40 00000000 c023bb6e d51a3980 00000000
       d51a3980 d51ac820 c023b9f2 d51a3980 d51a3980 c02d94a0 d537ec00 c022377e
       d537ec00 c037d340 d51a3980 d537ec00 c037d09c 00000000 c030ff80 c0224cc1
Call Trace:
 [<c023bb6e>] ip_local_deliver+0x153/0x1cd
 [<c023b9f2>] ip_rcv+0x37b/0x3a4
 [<c022377e>] netif_receive_skb+0x2a4/0x307
 [<c0224cc1>] process_backlog+0x6e/0xd3
 [<c0224e12>] net_rx_action+0x6d/0x139
 [<c0118563>] __do_softirq+0x35/0x75
 [<c01185c5>] do_softirq+0x22/0x26
 [<c0105074>] do_IRQ+0x48/0x50
 [<c0103a9a>] common_interrupt+0x1a/0x20
 [<c0101a51>] mwait_idle+0x20/0x33
 [<c0101a1c>] cpu_idle+0x37/0x4c
 [<c03105fa>] start_kernel+0x270/0x272
Code: 4b 18 0f 94 c0 84 c0 74 07 89 d8 e8 c5 12 cb e9 8b 46 20 8b 40 0c 0f c8 89
 04 24 66 c7 04 24 00 00 0f b6 54 24 07 0f b6 44 24 08 <0f> b6 00 50 0f b6 02 50
 68 f5 d5 56 d6 e8 8b 8f ba e9 83 c4 0c
EIP: [<d656c0c5>] mep_rcv+0xc5/0x194 [mep] SS:ESP 0069:c030ff04
 <0>Kernel panic - not syncing: Fatal exception in interrupt
_
```