SAMSUNG

Ethtool - Diagnostic Approach for Network Issues in Linux

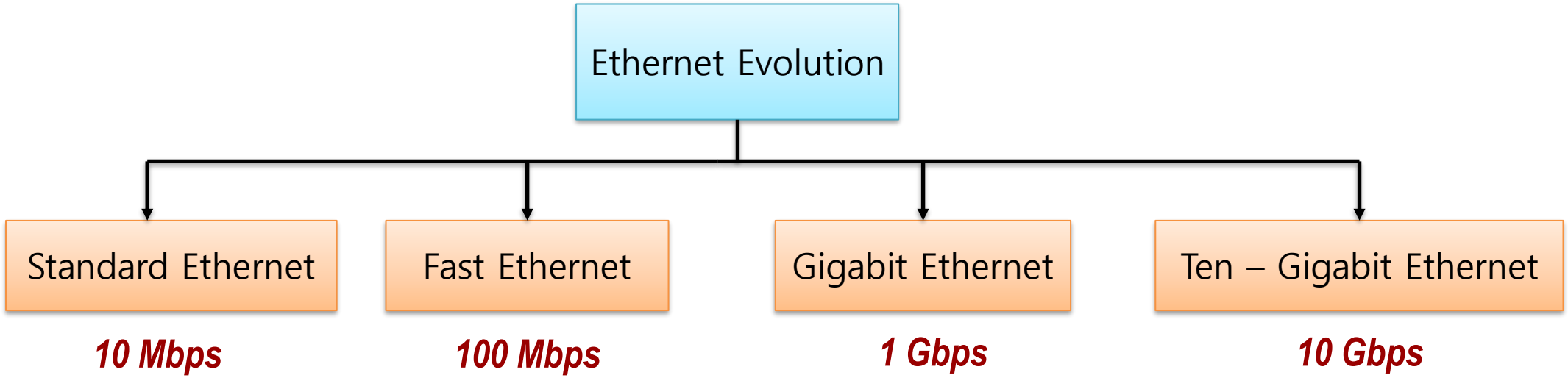
Sriranjani P, Ravi Patel

Samsung Semiconductor India R&D Center

Agenda

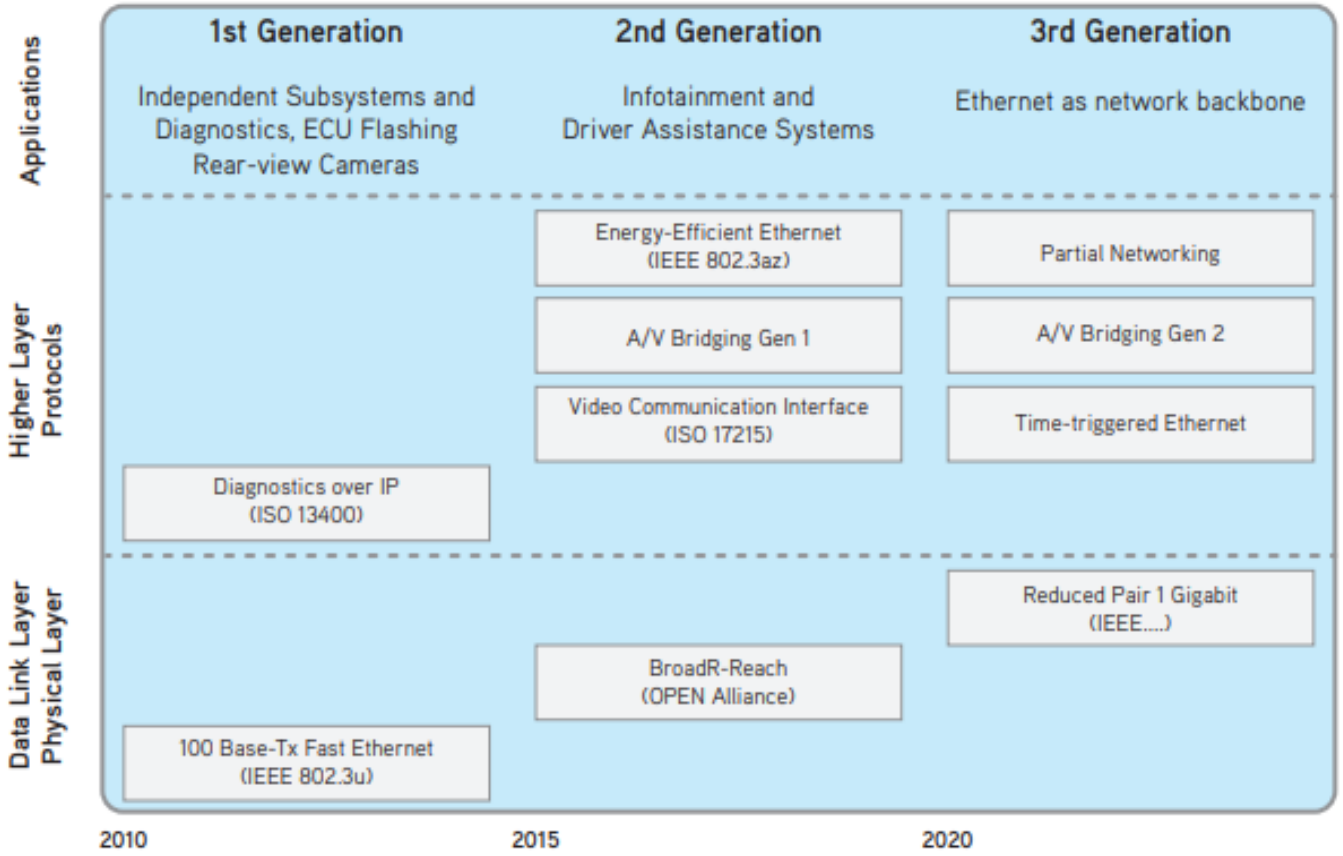
- Introduction to Ethernet
- Need of debugging and tuning of network
- About MAC Management Counters
- Why Ethtool
- Our Work
- Real Use - Case
- Results – Use case 1
- Results – Use case 2
- Conclusion and Future scope

- Standard Ethernet



Introduction to Ethernet

- Automotive Ethernet



(Image source: support.ixiacom.com)

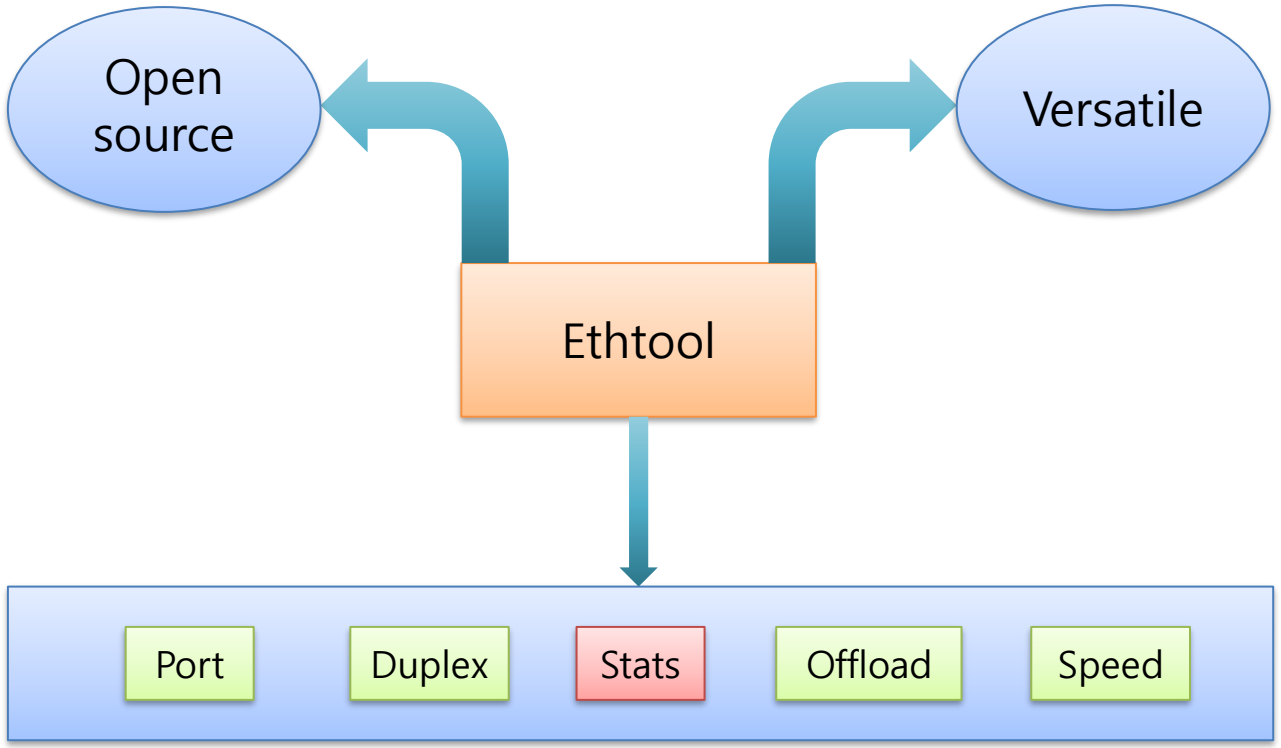
Need of debugging and tuning of Network

- To obtain information about network interface
- To validate the target host is accessible from our machine
- To get information regarding interrupts occurred during data transmission
- To find whether packet got dropped and what error occurred
- To ensure speed and mode of operation is proper
- Different tools available for various purpose like ping, ifconfig, arp, route, ethtool etc.

About MAC Management Counters

- Extension of the register address space of the CSR module
- Contains set of registers for gathering statistics on the received and transmitted packets
- Includes a control register for controlling the behavior of the registers

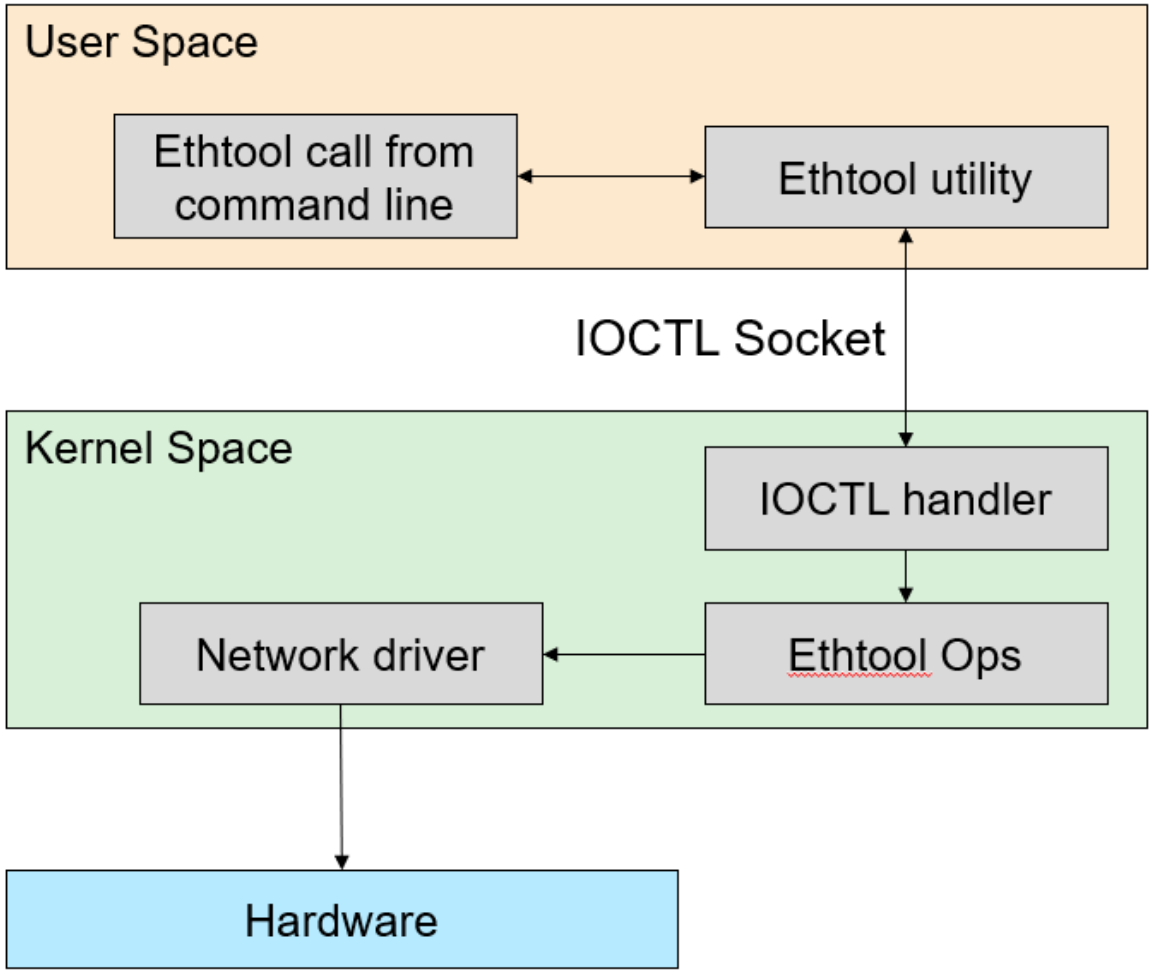
Why Ethtool


Open source
Versatile
Ethtool
Port
Duplex
Stats
Offload
Speed

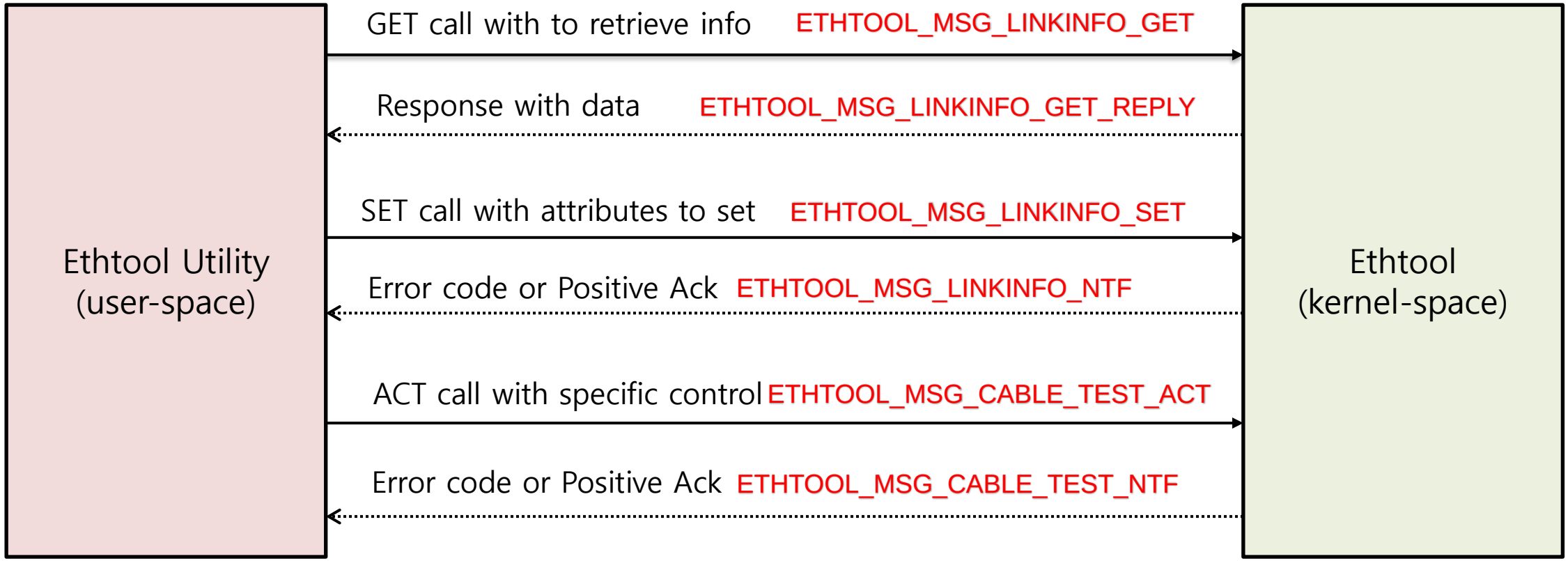
How to install Ethtool

- It can be used in most Linux distributions
- First update APT package
  - $sudo apt update
- Now install ethtool using,
  - $sudo apt install ethtool –y
- To install ethtool in Fedora/CentOS systems use,
  - $sudo dnf install python3-ethtool

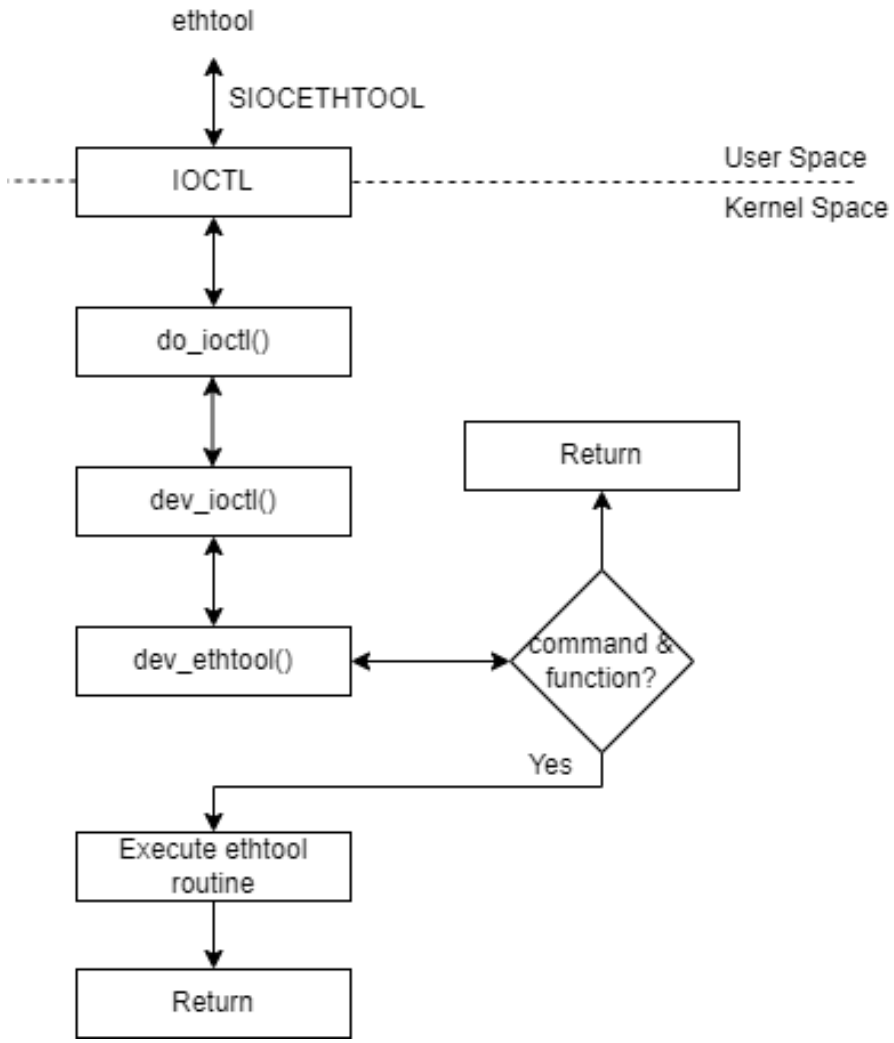
Ethtool Architecture Overview



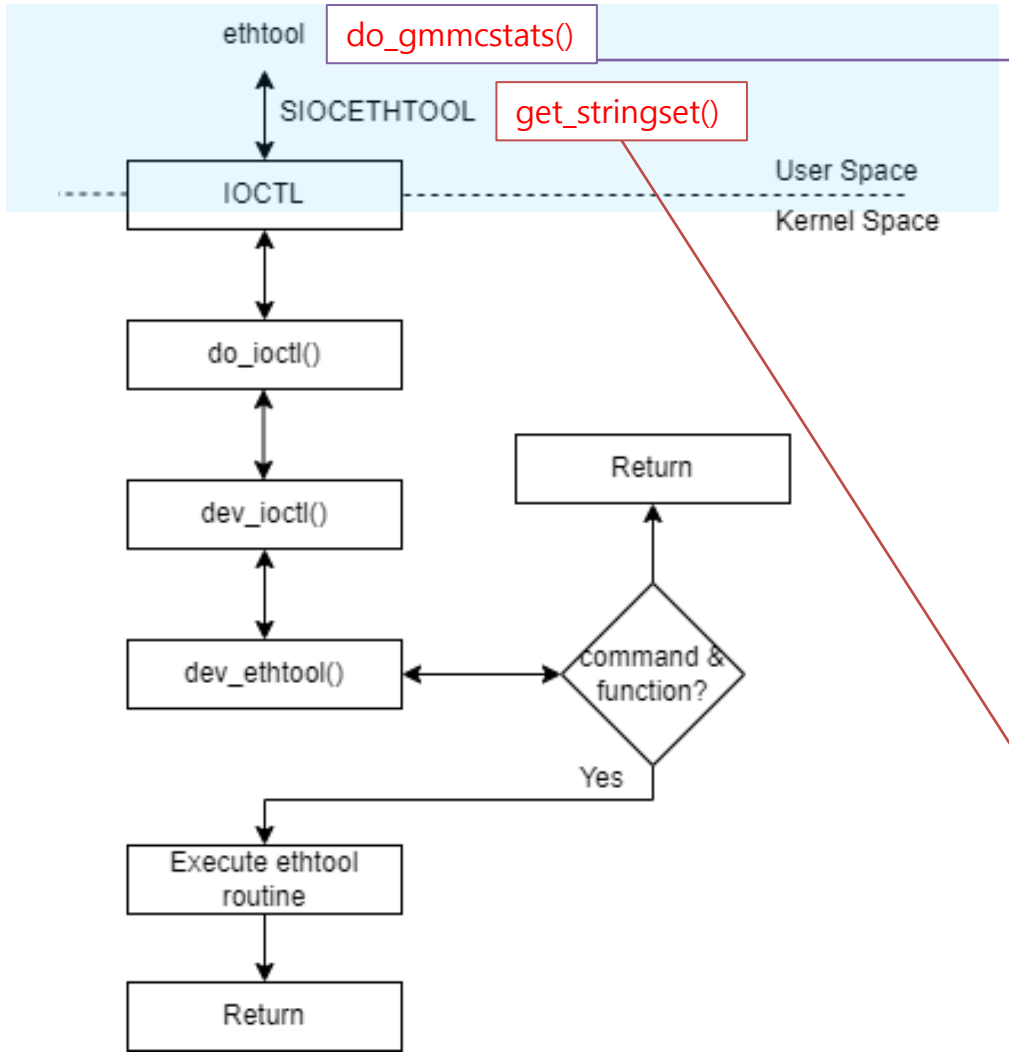
Kernel Infrastructure details of Ethtool



Ethtool IOCTL Interface Flowchart



Ethtool User space Implementation




```
{
    .opts   = "--show-mmc",
    .func   = do_gmmcstats,
    .help   = "Show DWC EQoS MMC register details",
},
```

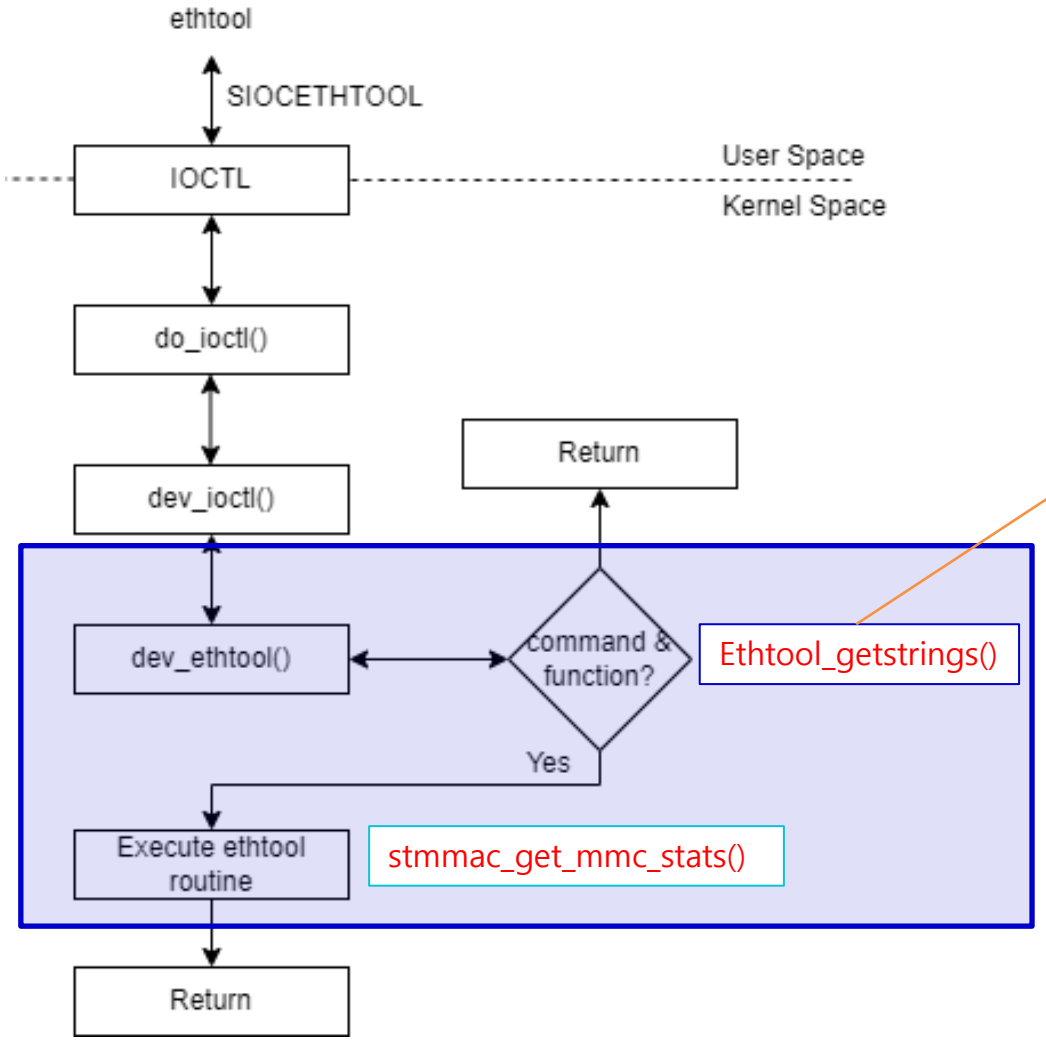
```
static int do_gmmcstats(struct cmd_context *ctx)
{
    return do_gmmc(ctx, ETHTOOL_MMC_STATS, ETH_SS_MMC_STATS, "MMC");
}
```

```
static int do_gmmc(struct cmd_context *ctx, int cmd, int stringset,
                   const char *name)
{
    struct ethtool_gstrings *strings;
    struct ethtool_eqos_mmc_stats *stats;
    unsigned int i;
    int err;

    if (ctx->argc != 0)
        exit_bad_args();

    strings = get_stringset(ctx, stringset,
                  offsetof(struct ethtool_drvinfo, n_stats),
                  0);
```

Ethtool Linux Kernel Implementation




```
switch (stringset) {
case ETH_SS_MMC_STATS:
    stmmac_get_mmc_stats(priv, p);
    break;
```

Ethtool_getstrings()

Ethtool Linux kernel Implementation


```
struct dwc_mmc_reg_definitions {
    int index;
    const char *name;
};
```

```
static struct dwc_mmc_reg_definitions s_dwc_mmc_cntrl_reg_definitions[BIT_COUNT] = {
{0, "CNTRST Counter Reset "},
{1, "CNTSTOPRO Counter Stop Rollover "},
{2, "RSTONRD Reset on Read "},
{3, "CNTFREEZ MMC Counter Freeze "},
{4, "CNTPRST Counters Preset "},
{5, "CNTPRSTLVL Full Half Preset "},
{8, "UCDBC Update MMC Counters for dropped Broadcast packets "},
};
```

```
static int bit[32];
static int dec_to_bin(int val)
{
    int index;

    for (index = 0; val > 0; index++) {
        bit[index] = val % 2;
        val = val / 2;
    }

    return 0;
}
```

Ethtool Linux kernel Implementation


```
static void stmmac_get_mmc_stats(struct stmmac_priv *priv, u8 *data)
{
    u32 val;
    int i,j=0;
    int offset = 0;
    char sample_buf[10000];

    val = readl(priv->ioaddr + DWCEQOS_MMC_CTRL);
    offset += sprintf(sample_buf + offset, "\tMMC Control val: 0x%x\n", val);
    dec_to_bin(val);
    if (bit[s_dwc_mmc_cntrl_reg_definitions[0].index] == 1)
        offset += sprintf(sample_buf + offset, "%s: Enabled\n", s_dwc_mmc_cntrl_reg_definitions[0].name);
    if (bit[s_dwc_mmc_cntrl_reg_definitions[0].index] == 0)
        offset += sprintf(sample_buf + offset, "%s: disabled\n", s_dwc_mmc_cntrl_reg_definitions[0].name);
    if (bit[s_dwc_mmc_cntrl_reg_definitions[1].index] == 1)
        offset += sprintf(sample_buf + offset, "%s: Enabled\n", s_dwc_mmc_cntrl_reg_definitions[1].name);
    if (bit[s_dwc_mmc_cntrl_reg_definitions[1].index] == 0)
        offset += sprintf(sample_buf + offset, "%s: disabled\n", s_dwc_mmc_cntrl_reg_definitions[1].name);
    if (bit[s_dwc_mmc_cntrl_reg_definitions[2].index] == 1)
        offset += sprintf(sample_buf + offset, "%s: Enabled\n", s_dwc_mmc_cntrl_reg_definitions[2].name);
    if (bit[s_dwc_mmc_cntrl_reg_definitions[2].index] == 0)
        offset += sprintf(sample_buf + offset, "%s: disabled\n", s_dwc_mmc_cntrl_reg_definitions[2].name);
    if (bit[s_dwc_mmc_cntrl_reg_definitions[3].index] == 1)
        offset += sprintf(sample_buf + offset, "%s: Enabled\n", s_dwc_mmc_cntrl_reg_definitions[3].name);
    if (bit[s_dwc_mmc_cntrl_reg_definitions[3].index] == 0)
        offset += sprintf(sample_buf + offset, "%s: disabled\n", s_dwc_mmc_cntrl_reg_definitions[3].name);
    if (bit[s_dwc_mmc_cntrl_reg_definitions[4].index] == 1)
        offset += sprintf(sample_buf + offset, "%s: Enabled\n", s_dwc_mmc_cntrl_reg_definitions[4].name);
    if (bit[s_dwc_mmc_cntrl_reg_definitions[4].index] == 0)
        offset += sprintf(sample_buf + offset, "%s: disabled\n", s_dwc_mmc_cntrl_reg_definitions[4].name);
```

Real Use - Case

- On TX side ifconfig shows these errors,

```
TX packets:2009 errors:1360 dropped:0 overruns:792 carrier:574
```

- Ping flood test with difference packet size shows no packet drops but the error sta ts increases for each packet when size is more than 100bytes
- Disable Transmit Status in MTL was set – this caused issue in updating status field of TX descriptor
- Unable to find what error occurred, does it mean S/W won't be able to find what all transmission error occurred??

Results – Use Case - 1

- Use Case 1:
  - mmc_cntrl register value is not show in existing ethtool statistics command which have bit-wise information useful for debugging purpose
  - We added additional support for mmc_cntrl register so that user gets more details information about this critical register information in user space
  - Out put from modified “ethtool” command is shown below:

```
# ./ethtool --show-mmc eth0

MMC Control val: 0x24
    CNTRST Counter Reset : Disable
    CNTSTOPRO Counter Stop Rollover : Disable
    RSTONRD Reset on Read : Enable
    CNTFREEZ MMC Counter Freeze : Disable
    CNTPRST Counters Preset : Disable
    CNTPRSTLVL Full Half Preset : Enable
    UCDBC Update MMC Counters for dropped Broadcast packets : Disable
```

Results – Use Case - 2

- Use Case 2:
  - mmc_rx_ipc_intr_mask value is shown as a big number in existing ethtool command like 1073692671
  - It's difficult for user to decode to get bitwise information

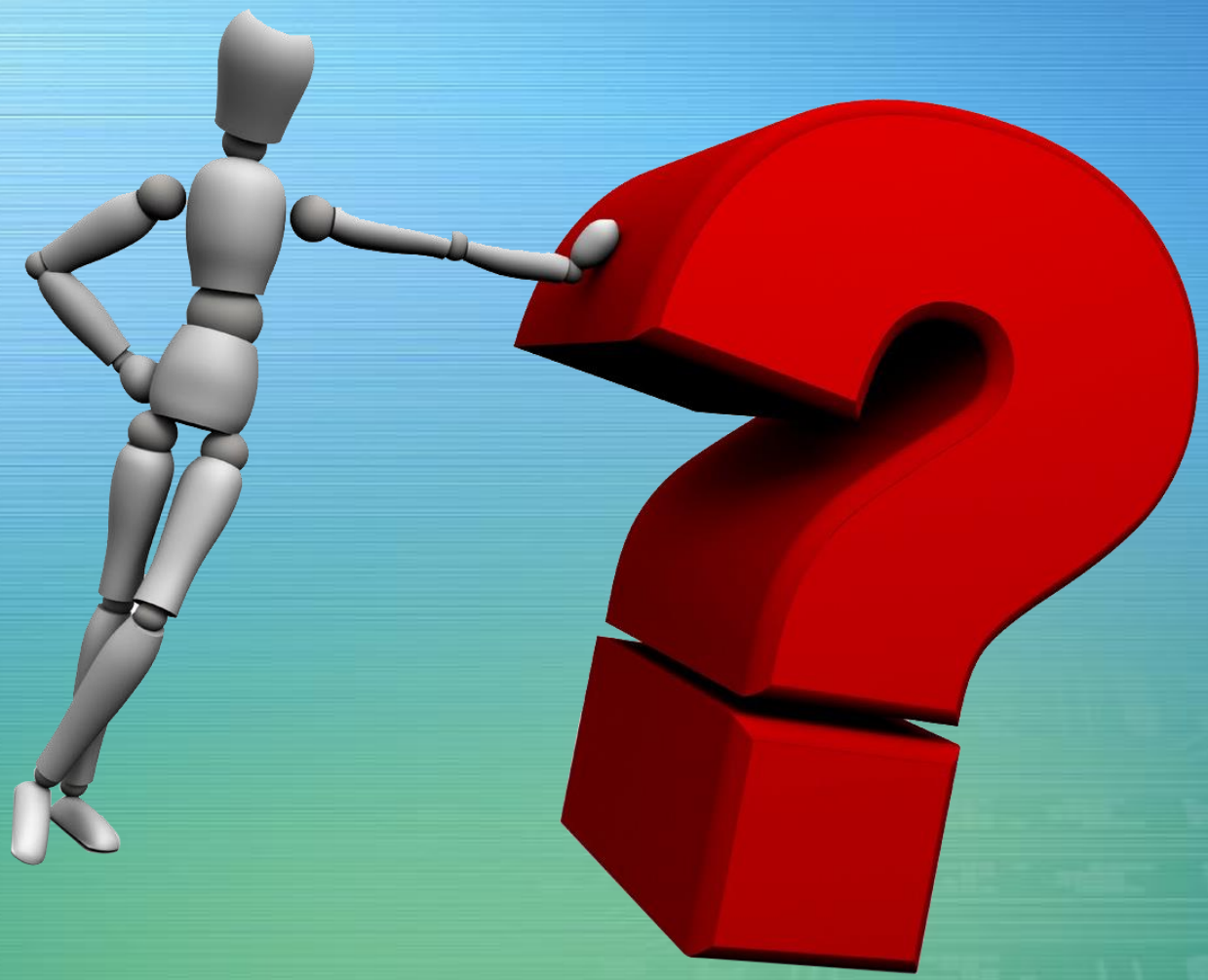
```
# ./ethtool --show-mmc eth0

MMC IPC Rx Interrupt Mask val: 0x3fff3fff
     RXIPV4GPIM MMC Receive IPV4 Good packet counter Interrupt Mask : Enabled
     RXIPV4HERPIM MMC Receive IPV4 Header Error packet counter Interrupt Mask : Enabled
     RXIPV4NOPAYPIM MMC Receive IPV4 No payload packet counter Interrupt Mask : Enabled
     RXIPV4FRAGPIM MMC Receive IPV4 Fragmented packet counter Interrupt Mask : Enabled
     RXIPV4UDSBLPIM MMC Receive IPV4 UDP Checksum Disabled packet counter Interrupt Mask : Enabled
     RXIPV6GPIM MMC Receive IPV6 Good packet counter Interrupt Mask : Enabled
     RXIPV6HERPIM MMC Receive IPV6 Header Error packet counter Interrupt Mask : Enabled
     RXIPV6NOPAYPIM MMC Receive IPV6 No Payload packet counter Interrupt Mask : Enabled
     RXUDPGPIM MMC Receive UDP Good packet counter Interrupt Mask : Enabled
     RXUDPERPIM MMC Receive UDP Error packet counter Interrupt Mask : Enabled
     RXTCPGPIM MMC Receive TCP Good packet counter Interrupt Mask : Enabled
     RXTCPERPIM MMC Receive TCP Error packet counter Interrupt Mask : Enabled
     RXICMPGPIM MMC Receive ICMP Good packet counter Interrupt Mask : Enabled
     RXICMPERPIM MMC Receive ICMP Error packet counter Interrupt Mask : Enabled
     RXIPV4GOIM MMC Receive IPV4 Good octet counter Interrupt Mask : Enabled
     RXIPV4HEROIM MMC Receive IPV4 Header Error octet counter Interrupt Mask : Enabled
     RXIPV4NOPAYOIM MMC Receive IPV4 No Payload octet counter Interrupt Mask : Enabled
     RXIPV4FRAGOIM MMC Receive IPV4 Fragmented octet counter Interrupt Mask : Enabled
     RXIPV4UDSBLOIM MMC Receive IPV4 UDP Checksum Disabled octet counter Interrupt Mask : Enabled
     RXIPV6GOIM MMC Receive IPV6 Good octet counter Interrupt Mask : Enabled
     RXIPV6HEROIM MMC Receive IPV6 Header Error octet counter Interrupt Mask : Enabled
     RXIPV6NOPAYOIM MMC Receive IPV6 No Payload octet counter Interrupt Mask : Enabled
     RXUDPGOIM MMC Receive UDP Good octet counter Interrupt Mask : Enabled
     RXUDPEROIM MMC Receive UDP Error octet counter Interrupt Mask : Enabled
     RXTCPGOIM MMC Receive TCP Good octet counter Interrupt Mask : Enabled
     RXTCPEROIM MMC Receive TCP Error octet counter Interrupt Mask : Enabled
     RXICMPGOIM MMC Receive ICMP Good octet counter Interrupt Mask : Enabled
     RXICMPEROIM MMC Receive ICMP Error octet counter Interrupt Mask : Enabled
```

Conclusion and Future Scope

- We are able to enhance “ethtool” user space utility to capture some of the critical debug registers of DWC EQoS controller and present the data to user-space in human readable format instead of just dumping register values
- We modified the DesignWare EQoS driver (drivers/net/ethernet/stmicro/stmmac/stmmac_ethtool.c) in kernel by adding required ethtool_ops hooks to get all MMC register value, bit wise information of mmc_cntrl, interrupt status and interrupt mask register defined in DWC EQoS IP
- This implementation helping us to get these information via ethtool command line
- Future scope involves implementing this functionality in ethtool for other MMC registers and upstreaming the source code to the Linux mainline community

Any Questions ?



THANK YOU