Ministry of Education and Science of Ukraine

State University of Telecommunications

Control plane requirements for wireless and cellular networks based on SDN

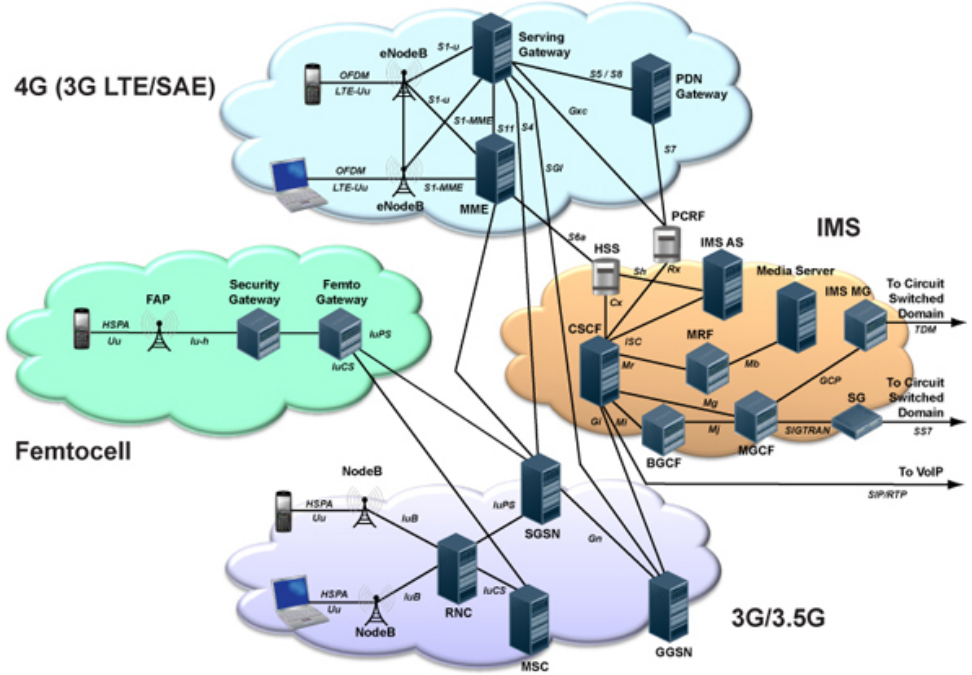
Pavlo Melnikov, Ph.D. student
State University of Telecommunications

Content

Introduction

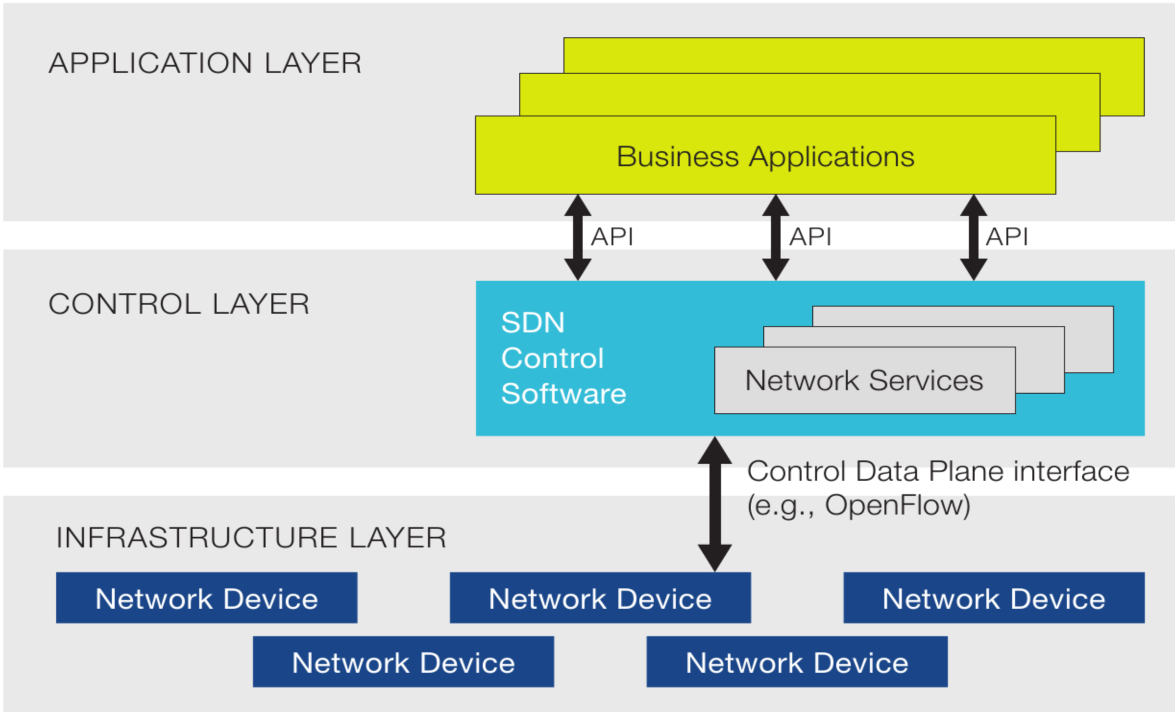
Problems with Current Cellular Networks:

- Plagued by complex and inflexible architecture
- Most data plane related functionality centralized
- Control plane too distributed e.g., radio resource allocation
- No clear separation between control and data planes



Too complex and inflexible

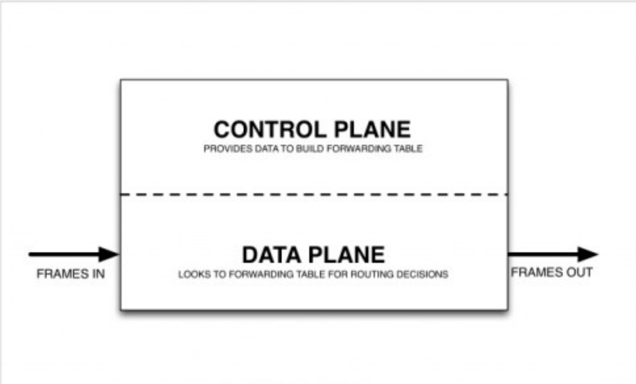
SDN Architecture



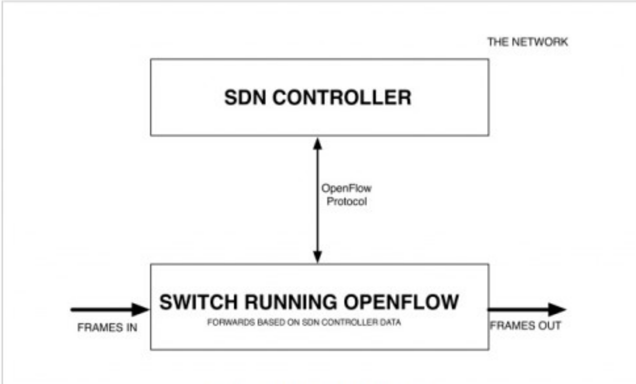
Simplifying complexity in network control

Main terminology of SDN

- OpenFlow is the protocol which is used in software defined networks for communicating between SDN operational nodes. Can be used as a control and management protocol
- Control Plane is the layer of decisions applying an incoming and outgoing traffic through the network. Control plane packets are processed by the router to update the routing table information
- Data Plane is the layer which forwards traffic to the next hop along the path to the selected destination network according to control plane logic



Traditional Networking



Software Defined Networking

Appliance of SDN in wireless and cellular networks

- Offers a logically centralized control plane - will lead to simpler and effective radio resource management (e.g., inter-cell interference management)
- Enables common control protocol across diverse wireless technologies - will ease seamless mobility support within and across technologies (e.g., 4G LTE, 3G UMTS, WiFi)
- Allows distributing traffic monitoring at switches deep inside the core network and ease the burden on the packet gateway

SDN and Openflow Use Case

Appliance of SDN in control plane for receiving statistics data from wireless and cellular networks on:

- Physical (RSSI, Antenna Gain, Throughput),
- Data Link (MAC Addresses, Frame Counters),
- Network (IP addresses, subnetworks, Packet Counters),
- Transport (active TCP/UDP connections)
- Session (Number of active sessions (VoIP))

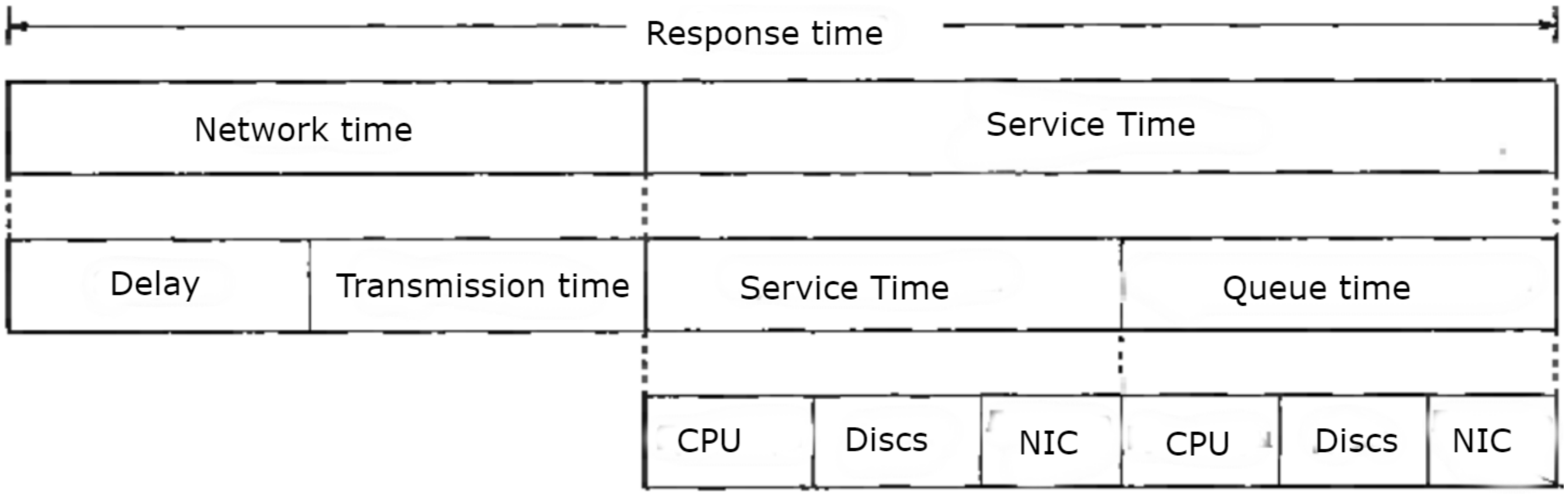
layers;

Requirements to the system

Requirements to a control plane

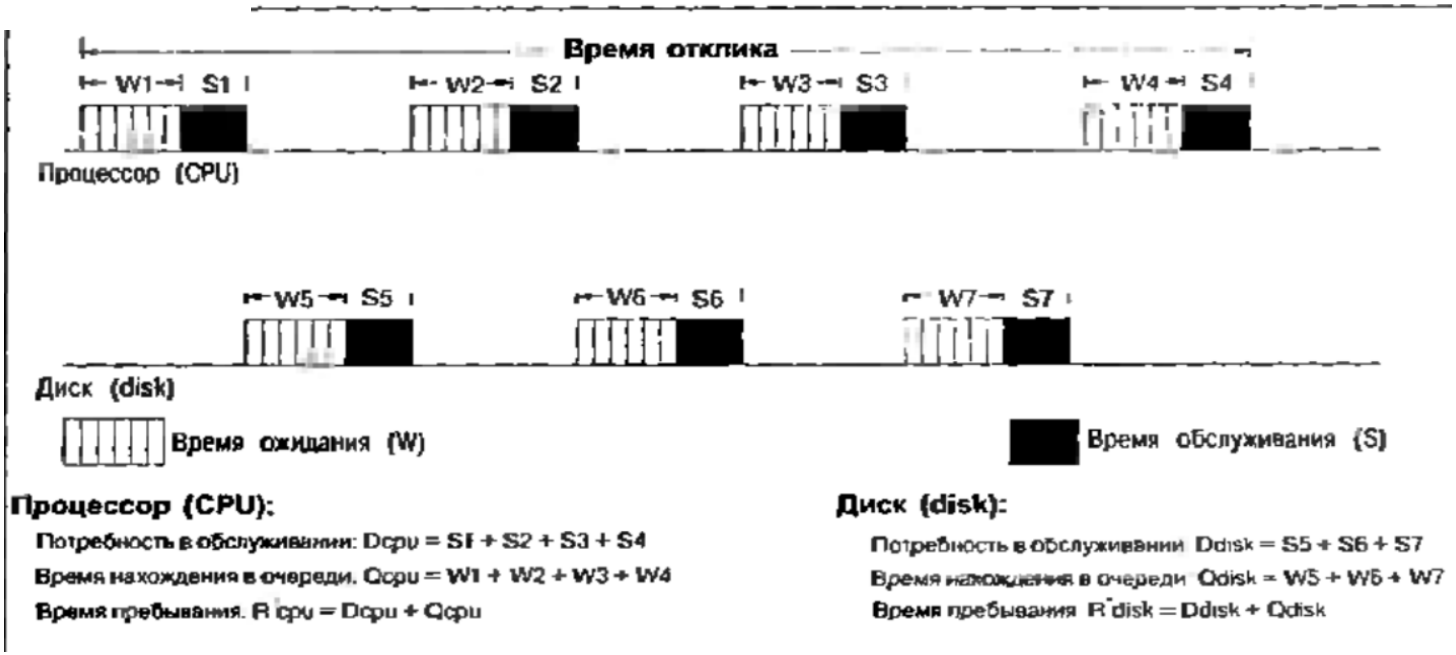
- Response delay minimization
- Statistics gathering on enterprise servers for further analysis by administrator/system
- Scaling ability at increasing the number of end devices in a network

Request-Response delay



Parts of a response delay

Response delay minimization



To minimize response delay, we need to define bandwidth server and router delay

Response delay minimization

The goal is to define Bandwidth-delay
minimization which is introduced by a TCP Protocol

Bandwidth x Delay = BDP

Server delay

The goal is to define delay minimization introduced by a RAID5 disc array which is located on a server.

After counting the number of read operations, which are made by read write requests, we can get an equation for an input intensity of disc requests

$$\lambda_{disk}^{r} = \frac{n_r}{N} \times \lambda_{array}^{r} + \frac{rw(n_w)}{N} \times \lambda_{array}^{w}$$

Input intensity for request of one server disc

$$\lambda_{disk}^{r} = (4/5 \times \lambda_{array}^{r})/2 + (1/5 \times \lambda_{array}^{r})/2 = \lambda_{array}^{r}/2$$

Network delay

The goal is to define delay minimization which is introduced by a border router.
The time nedded for router for datagram processing is called router delay and its value usually is represented by router manufacturer. The general time of message servicing on a router:

$$Rts = Nd \times Tdr$$

, where

Rts – Router service time
Nd – number of datagrams
Tdr – Router delay (latency)

Use Case: network equipment statistics request using Openflow protocol

Request

```
switch(reqinfo->mode)
 {
  case MODE_GET:
   pthread_mutex_lock(&cambstats_mutex);
   switch(reginfo->handlerName[0])
   {
    case '0': // 0 force_upd
     camb_stats_schedule_update(STATS_UPD_ATH_ETH);
     break;

    case '1': // 1 eth_rx_bytes
     param_64 = eth0_stats_res.rx_bytes >> 7; break;    /*convert to KBit 8/1024 /
    case '2': // 2 eth_rx_packets
     param_64 = eth0_stats_res.rx_packets; break;
    case '3': // 3 eth_rx_errors
     param_64 = eth0_stats_res.rx_errors; break;
    case '4': // 4 eth_rx_drops
     param_64 = eth0_stats_res.rx_drops; break;
    case '5': // 5 eth_rx_multicast
     param_64 = eth0_stats_res.rx_multicast; break;
    case '6': // 6 eth_rx_broadcast
     param_64 = eth0_stats_res.rx_broadcast; break;
    case '7': // 7 eth_tx_bytes
     param_64 = eth0_stats_res.tx_bytes >> 7; break;    /*convert to KBit 8/1024 /
    case '8': // 8 eth_tx_packets
     param_64 = eth0_stats_res.tx_packets; break;
    case '9': // 9 eth_tx_errors
     param_64 = eth0_stats_res.tx_errors; break;
    case 'A': // A eth_tx_drops
     param_64 = eth0_stats_res.tx_drops; break;
    case 'B': // B eth_tx_multicast
     param_64 = eth0_stats_res.tx_multicast; break;
}
```

Response

```
root@serial79: ~ 94x21
   "FrameSkipDueToCCACounter": 0,
   "cambiumStatsResetTimer": "0001:13:43:52"
}
root@ePMP1000_c6f6cf:/usr/share/udhcpc# tddstats eth
{
   "rxEtherLanKbitCount": 124175,
   "rxEtherLanTotalPacketCount": 49137,
   "rxEtherLanErrorPacketCount": 0,
   "rxEtherLanDroppedPacketCount": 0,
   "rxEtherLanMulticastPacketCount": 82,
   "rxEtherLanBroadcastPacketCount": 2110,
   "rxEtherLanMultiBroadcastKbitCount": 2949,
   "txEtherLanKbitCount": 305240,
   "txEtherLanTotalPacketCount": 315474,
   "txEtherLanErrorPacketCount": 0,
   "txEtherLanDroppedPacketCount": 0,
   "txEtherLanMulticastPacketCount": 42967,
   "txEtherLanBroadcastPacketCount": 268285,
   "txEtherLanMultiBroadcastKbitCount": 270787
  }
root@ePMP1000_c6f6cf:/usr/share/udhcpc#
```

Conclussion

Using SDN control plane for gathering and analyzing statistics data of terminal equipment, calculating the intensity of read requests disk arrays, reducing the request-response latency is beneficial for using in wireless and cellular networks.

Thank You!