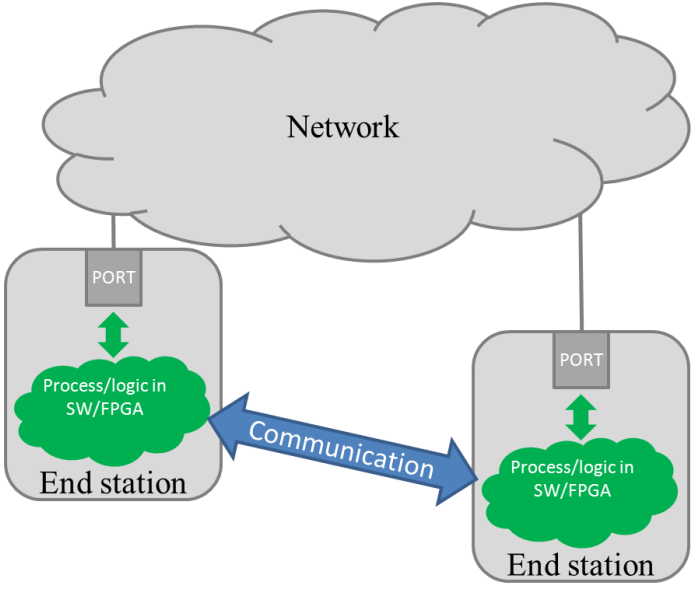
# Overview of relevant network protocols and standards

Maciej Lipinski
BE-CEM-EDL

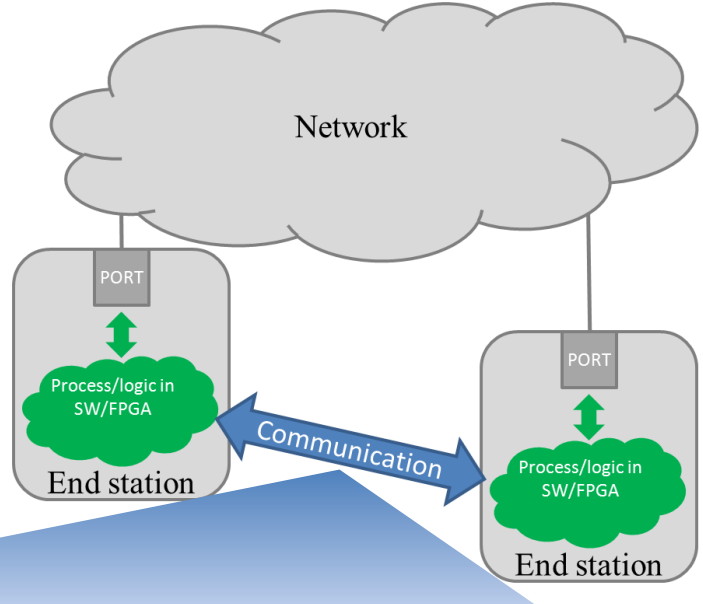
# Discussion on protocols

- Goal: transmit user application-specific data



# Discussion on protocols

- Goal: transmit user application-specific data
- Protocol of choice
  - Application-specific (more or less)
  - Provides structure to the user data
  - Complements underlying transport protocol with required functionalities




Underlying protocol:

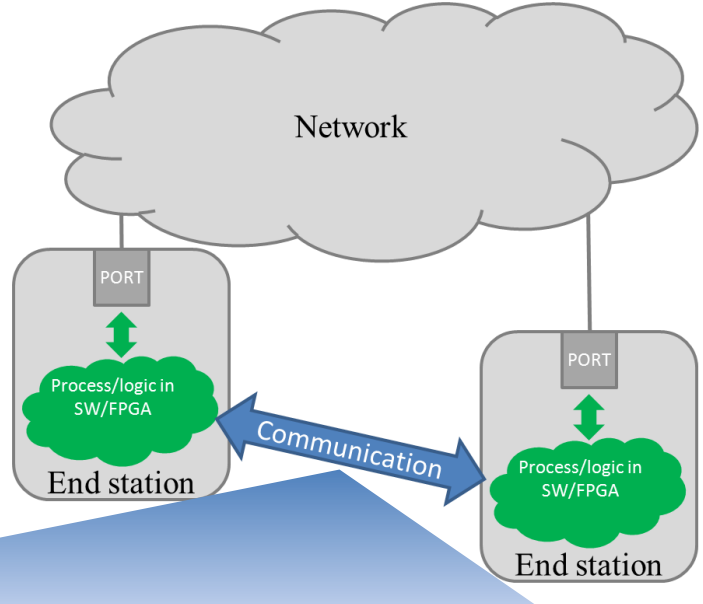
| Header(s) of underlying protocol(s) | Payload with protocol of choice | CRC |
|---|---|---|

Protocol of choice:

| Header of protocol of choice | Structured user data |
|---|---|

# Discussion on protocols

- Goal: transmit user application-specific data
- Protocol of choice
  - Application-specific (more or less)
  - Provides structure to the user data
  - Complements underlying transport protocol with required functionalities
- Underlying protocol(s) → **focus of this presentation**
  - Addressing and abstract physical network
  - Services/facilities (checksum, reliability)
  - Application agnostic
  - Suited to particular applications




Underlying protocol:

| Header(s) of underlying protocol(s) | Payload with protocol of choice | CRC |
|---|---|---|

Protocol of choice:

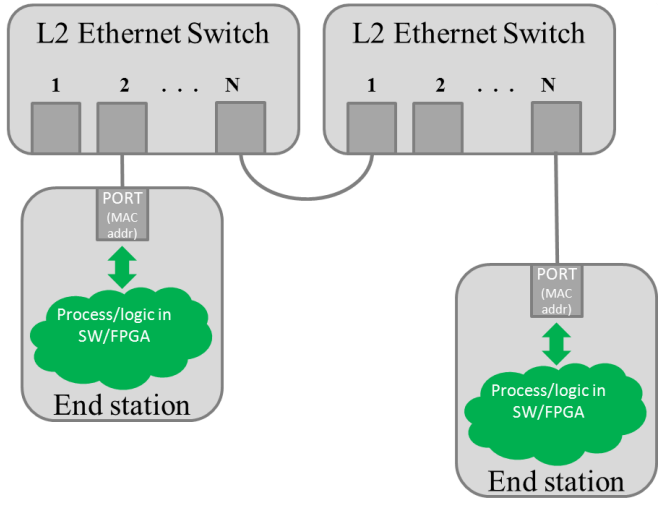
| Header of protocol of choice | Structured user data |
|---|---|

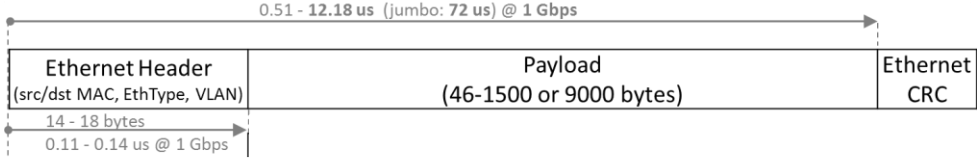
# Agenda

- Raw Ethernet
- UPD/IP
- TCP/IP
- Comparison
- Standard vs. custom protocol

# Raw Ethernet

- Raw Ethernet
  - Bridged Local Area Network (IEEE 802.1Q)
  - Ethernet medium (IEEE 802.3)
- Communication
  - Within LAN (L2 Switches only)
  - Ethernet Frames at OSI Layer 2: Data Link
- Addressing
  - Globally unique MAC/physical address
  - Pre-assigned (*)
- Example networks:
  - A simple WR network

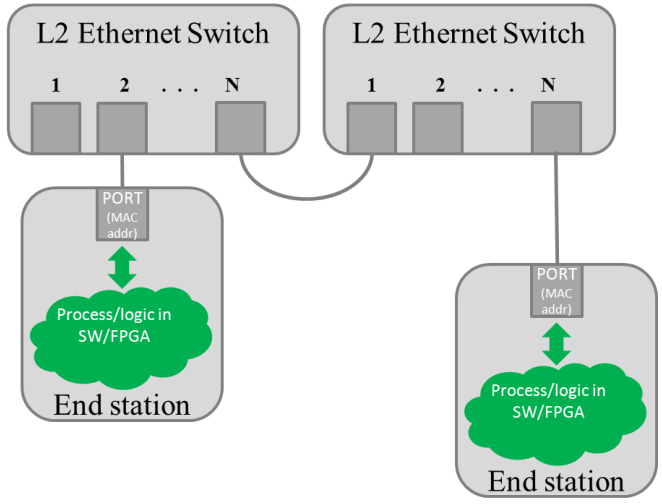


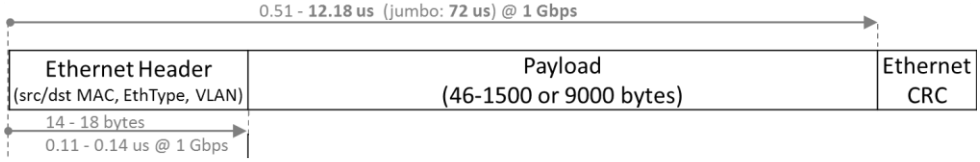




(*) Off the shelf devices have unique pre-assigned MAC addresses, it is less so for FPGA-based custom devices

# Transmission over Raw Ethernet

- Yes:
  - Checksum to verify data integrity
  - Broadcast/multicast
- No:
  - Application multiplexing (#)
  - Connection/handshake before sending data
  - Detection/retransmission of lost frames
  - Reordering of out-of-order frames
  - Congestion control (*)
  - Segmentation/fragmentation
- Simple stack implementation
  (on Linux, sudo required for transmission)
- Smallest header overhead
- Lowest latency
  - Tx: no buffering to calculate CRC
  - Forwarding: L2 Switches typically lower latency
  - Rx: no buffering

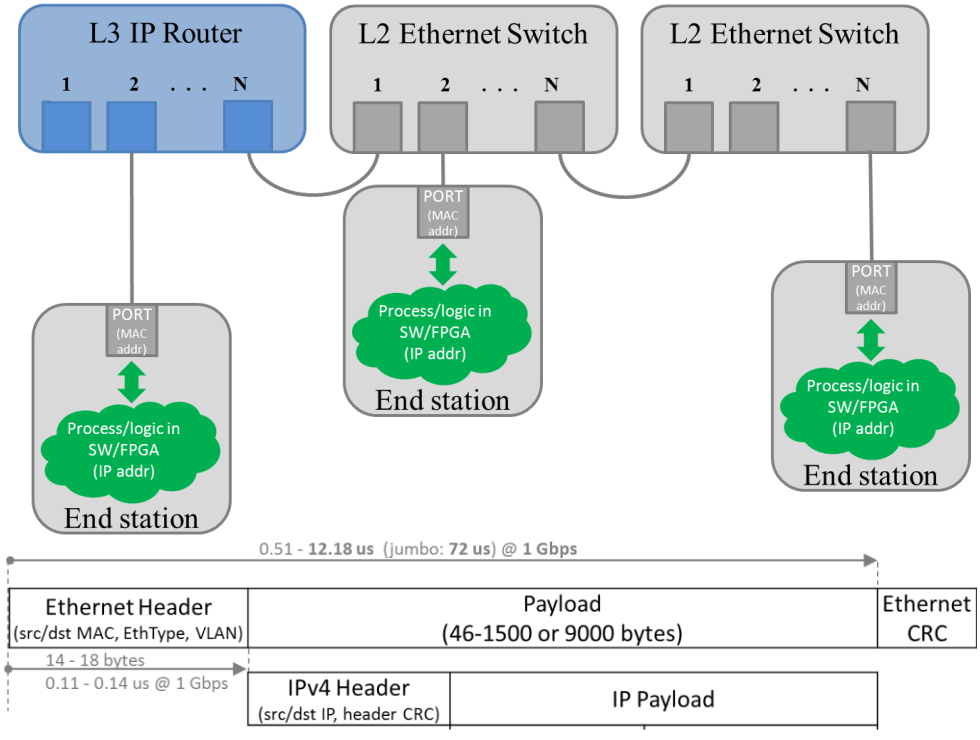



(#) WR streamers use ("illegally") EthType to do initial application multiplexing, another multiplexing is done inside Ethernet payload
(*) Congestion control via the Ethernet PAUSE mechanism. However, it is not embedded in the Ethernet Frame.

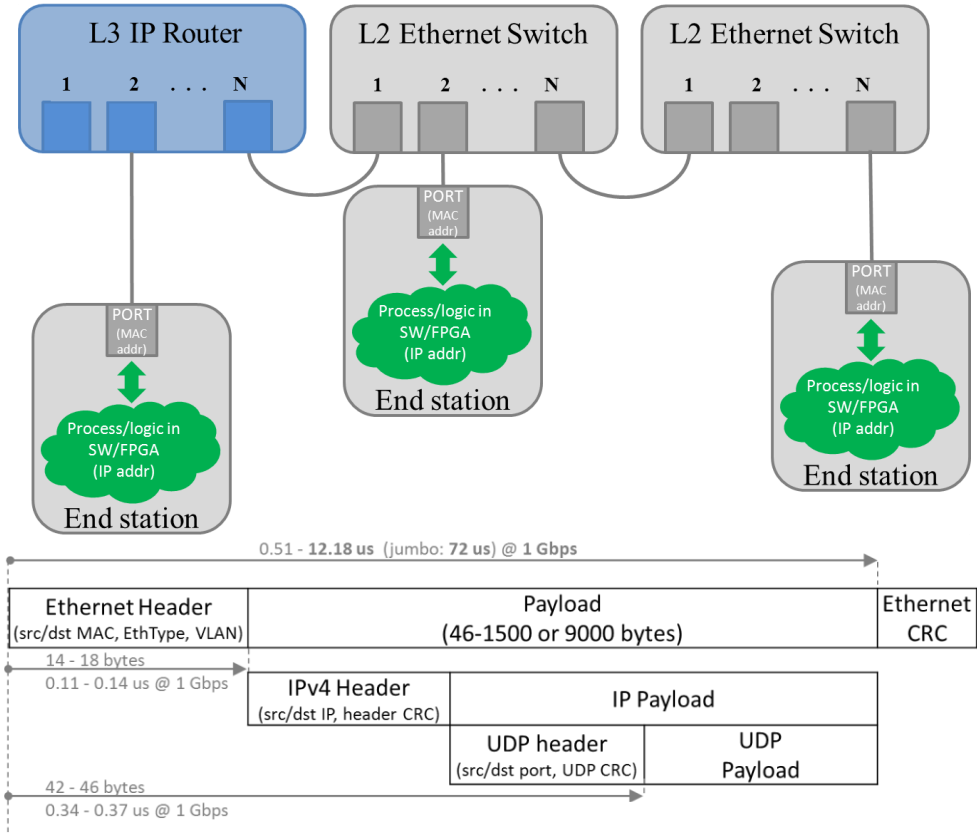
# UDP and TCP over IP

- Internet Protocol
  - Version 4 (IPv4): IETF RFC 791
  - Version 6 (IPv6): IETF RFC 2460
- Communication:
  - IP at OSI Layer 3: Network
  - UDP/TCP at OSI Layer 4: Transport
  - Within and outside LAN (L2 Switches or L3 Routers)
- Addressing
  - Locally unique IP addresses
  - Manual or automatic assignment, e.g. DHCP server
- Max size of IP datagram: 65,535 bytes
- IPv4 supports fragmentation
  - Fragment to meet maximum transmission unit (MTU)
  - Reassemble and re-order
  - Discouraged, can be disabled
- Example network:
  - CERN Technical Network
  - CERN operational WR network



# Transmission over UDP/IP

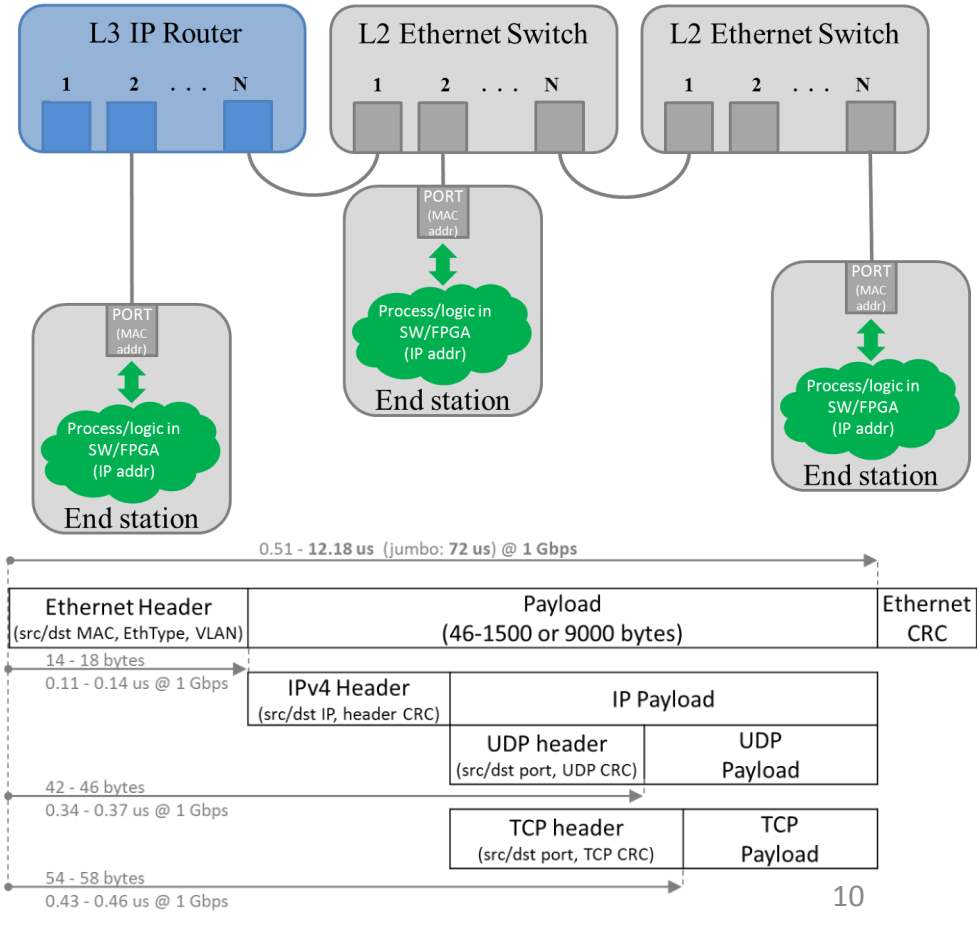
- User Datagram: RFC 768
- Yes:
  - Checksum to verify data integrity
  - Application multiplexing (port number)
  - Broadcast/multicast
- No:
  - Connection/handshake before sending data
  - Detection/retransmission of lost frames
  - Reordering of out-of-order UDP datagrams (#)
  - Congestion control (*)
- Simple stack implementation
  (on Linux, transmission form user space)
- Still small header overhead
- Low latency but latency added
  - At tx due to CRC/length in the header
  - At routers due to additional headers
  - At routers/rx due to IP fragmentation, if enabled
  - L3 Routers typically slower than L2 Switches




(#) Support for fragmentation in the underlying IP
(*) Congestion control via the Ethernet PAUSE mechanism within VLAN. However, it is not embedded in the UDP packet.

# Transmission over TCP/IP

- Transmission Control Protocol: RFC 793
- Yes:
  - Checksum to verify data integrity
  - Application multiplexing (port number)
  - Connection/handshake before sending data
  - Detection/retransmission of lost frames
  - Reordering of out-of-order frames
  - Congestion control
- No:
  - Broadcast/multicast
- Complex/heavy stack implementation
  (on Linux, transmission form user space)
- Unpredictable/high latency due to
  - Connection
  - Retransmission
  - Reordering
  - Congestion control



# Comparison

| | Raw Ethernet | UDP/IP | TCP/IP |
|---|---|---|---|
| Addressing | MAC address pre-assigned | IP address assigned by user or server (e.g. DHCP) | IP address assigned by user or server (e.g. DHCP) |
| Application multiplexing | NO (#) | YES | YES |
| Integrity check (checksum) | YES | YES | YES |
| Broadcast/multicast | YES | YES | NO |
| Communication model | Connectionless | Connectionless | Connection-oriented |
| Initial handshake | NO | NO | YES |
| Reliable | NO | NO | YES |
| Lost frames detection/retransmissions | NO | NO | YES |
| Data order ensured | NO | NO (^) | YES |
| Congestion control | NO (*) | NO (*) | YES |
| Implementation complexity | LOW | LOW | HIGH |
| Latency | LOWEST | STILL LOW | UNPREDICTABLE/HIGH |

(#) WR streamers use (“illegally”) EthType to do initial application multiplexing, another multiplexing is done inside Ethernet payload
(^) Fragmentation and re-ordering is supported by the underlying IP
(*) Congestion control via the Ethernet PAUSE mechanism within VLAN. However, it is not embedded in the UDP packet.

# Standards vs. custom protocol of choice

(applies equally well for the underlying protocols)

| | Standard protocols | Custom protocols |
|---|---|---|
| (+) | • Likely implementation exists and is tested<br>• Likely debugging/testing tools exist already (e.g. Wireshark)<br>• Likely will evolve with underlying standards<br>• Off-the-shelf solutions available/compatible<br>• Easy to export/share<br>• Enforce generic solutions, avoid design mistakes<br>• Hard/long to incorporate improvements in the standards (stable !) | • Optimized precisely for the application/needs<br>• Typically a seemingly easier solution<br>• Easy to make improvements/changes |
| (-) | • Hard/long to incorporate improvements in the standards<br>• Generic, thus possibly more complex<br>• Usually not a perfect fit for the needs<br>• Legacy burden | • Easy to make improvements/changes &mistakes<br>• Harder to use outside a particular setup<br>• Easy to make a non-extensible/non-scalable solution<br>• Less likely to be adopted by others<br>• Hard to get external help in case of problems<br>• Harder/costly to outsource work<br>• Maintenance costs |

Thank you