

Application and Transport Layer Standards

*Muhammad Hazimi Bin Yusri (mhby1g21, Team E1),
Law Shaw Zuan (szllc21, Team E2)
ELEC3227 Embedded Networked Systems Coursework*

Application Layer (APP)

- Hostnames are E11, E12, E13 and E21, E22, E23 for Team E1 and E2.
- One LED on PC6 and one button on PA6, UART communication at Port B.
 - Leftover pins can be used for optional implementation by individuals.
- “IPPLS” request is broadcast together with device own hostname and IP address whenever a node joins a network to get other devices hostname and IP address. The receiving nodes replies with APP Data containing 3 bytes hostname and 1 byte IP address it is associated with.
- Hostname-IP address table is used to resolve hostname into destination IP address for Network Layer.

Application 1: Button interaction with LED

- Only 1 byte of APP data will be used.
- Button LED ON sends a byte of 0xFF, button LED OFF sends a byte of 0x00.
- Button/LED interactions between nodes are hard coded.
 - Button on a node will send data (LED ON/OFF) to only one of its adjacent nodes.
 - E.g.: E11 button will turn on E12 LED, E12 button will turn on E13 LED etc...
 - The ending node will turn on starting node (E23 button turns on E11 LED)

Application 2: UART as input and output for sending strings and specific LED commands

- UART for sending/receiving messages, constrained to 9 bytes or characters maximum.
 - E.g. string commands: SEND E11 <MESSAGES> (Send string messages to E11)
- UART LED commands use 9 bytes, the first left-most byte is reserved for basic ON/OFF byte functionality and the remaining bytes are for individual team use.
- UART commands to interact with LED on other nodes.
 - Example of basic commands: ON E11 (Turn on LED on node E11), OFF E23 (Turn off LED on node E23), ON E11, E22 (Turns on LED on E11, and E22)
 - Example of optional commands that uses additional parameters: ON E11 R5G9B7 (Turn on optional RGB LED with different brightness level given)
 - Clarification: UART to control LEDs will result in 0xFF source port and 0x00 LED port so it can be easily differentiated from button presses.
- The special character (“/”) is used to toggle between input or output mode in UART command line interface (cli).

Interfaces:

- APP Layer provides destination port, source port and destination IP address to TRAN Layer for correct connection.
- APP Layer provides APP data with correct length to TRAN Layer for segmentation.

Transport Layer (TRAN)

TRAN:	Control [2]	SRC Port [1]	DEST Port [1]	Length [1]	APP Data [1-114]	Checksum [2]
-------	----------------	-----------------	------------------	---------------	---------------------	-----------------

Figure 1. Segment field structures defined by the coursework

- Using TCP (3-way handshake connection) with segment structure shown below,
 - a. Control bits (2 bytes)

Left Byte Assignment

(15)	(14)	(13:11)	(10)	(9)	(8)
SYN	ACK	BUF	DR	IP_flag	IPPLS_flag

** SYN and ACK for handshake, BUF for buffer remaining, DR for disconnect request, IP_flag for signifying other device IP and hostname in 4-byte APP Data, IPPLS_flag to request other devices IP and hostname information.

Right Byte Assignment

(7:4)	(3:0)
SEQ NUM	ACK NUM

- b. SRC Port (1 byte)
 - 0x00 for button presses, 0xFF for UART cli inputs.
- c. DEST Port (1 byte)
 - 0x00 for LEDs, 0xFF for UART cli outputs.
- d. Length (1 byte)
 - Length of APP Data bytes which are not 0x00, for example during TCP initial and disconnect handshake, it would be 0 as no APP Data is sent.
 - Unused bytes of UART messages will be set to 0x00, thus will not always be 9 bytes.
- e. APP Data (0 bytes OR 1 byte OR 4 bytes OR 9 bytes)
 - 0 bytes for TCP handshake 1 byte for LED with button, 4 bytes for HOSTNAME|IP ADDRESS, and 9 bytes for UART messages/commands.
- f. Checksum (1 byte)
 - Even parity bit check
 - Simpler checksum is used to reduce the processing time requirement as both NET and DLL layer are already using CRC

Interfaces:

- TRAN Layer provides destination IP address to Network Layer.
- TRAN Layer provides segment to Network Layer

The absence of **Bandwidth Allocation** or **Congestion Control** is deliberate, due to the periodic and bursty nature of the data rate, to reduce microcontroller load for improved power efficiency and less overhead. In rare cases of congestion and collision, CSMA p-persistent from DLL combined with TCP connection-oriented transport should minimize its impact.