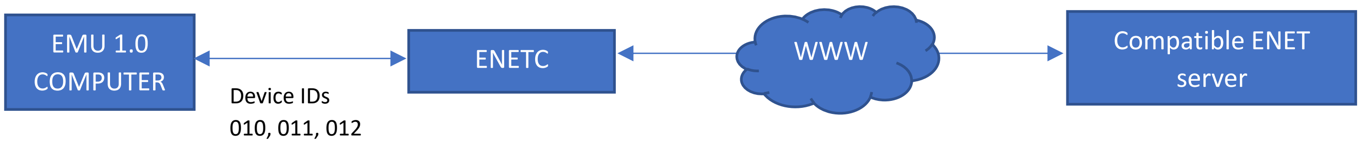
ELFTech Network interfaces – quick reference

The ENET Network interface family contains two physical devices for enabling EMU 1.0 compatible computers to communicate over TCP/IP in a client-server model.

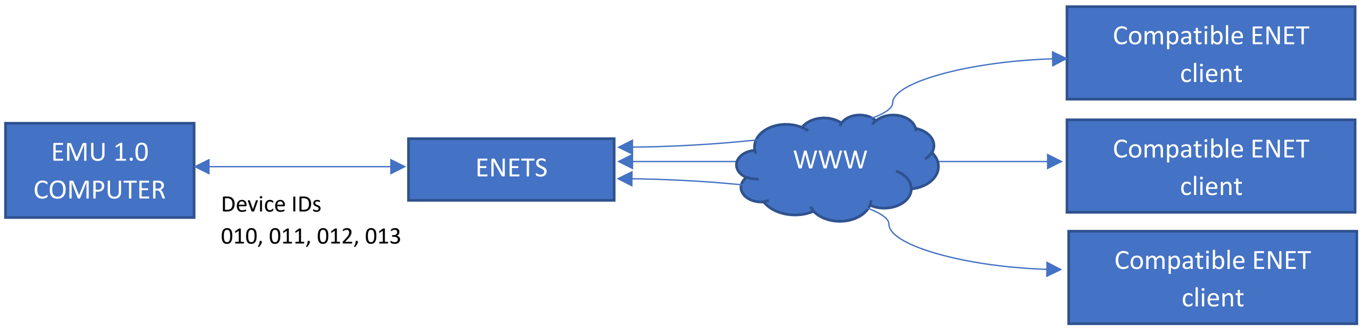
ENETC: ELFTech Network interface client

ENETC establishes a TCP/IP connection to a given server-host, selected via 48 physical DIP switches describing an IPv4 address and TCP port.



ENETS: ELFTech Network interface server

ENETS listens and manages a number of TCP/IP connections, allowing the controlling EMU 1.0 computer to cycle through active connections and communicate with compatible clients over them.



EMU 1.0 interface

Common exposed devices

Both ENETS and ENETC expose the following 3 devices to the EMU 1.0 I/O controller, which behave similarly to the standard Serial Port device:

Device 010: ENET_INCOMING – Reads the number of buffered words for this connection, clamped to max 63.

Device 011: ENET_RECV – Reads a buffered word, or -1 if there are none.

Device 012: ENET_SEND – Writes one word to the remote machine.

Any received byte greater than 63 is transformed to the “invalid” character, with code 63.

ENETS control device

ENETS has an internal connection circular queue to facilitate serving multiple clients, controlled via an additional device:

Device 013: ENET_CONN_CTRL – Write O to perform one of the following actions (unlisted O are ignored):

Set marker (O = 1) – Sets an internal marker to refer to the current connection, forgetting the previously marked connection. The marker does not refer to any connection on startup.

Cycle connection (O = 2) – Advances connection queue one step. Reads 1 if it either arrived at the marked connection, or the circular queue passed over where the marked connection would have been, in the case it disconnected since being marked. Reads 0 otherwise.

ENETS functionality description

The ENETS device opaquely handles connect and disconnect events, as well as buffering incoming data for all connections.

The internal connection queue may have more than 64 slots, which is why it is inappropriate to expose overall information about the connection queue to the controlling EMU 1.0 computer.

Upon receiving a connection, it is added to the end of the circular queue.

When cycling connections, the ones that have been closed by the client are repeatedly skipped and removed from the queue until an open connection is found. If any of the closed connections or the first open connection was the currently marked connection, the cycle command writes a 1 value.

Example server broadcast code

The following EMU 1.0 assembly marks the current position in the queue, then repeatedly sends the word 42 to each client until it loops back to the marked client, where it started.

```
  add r2, r0, 42

  # Set queue marker
  add r1, r0, 1
  io ENET_CONN_CTRL, r1

  lbl 01000
  io ENET_SEND, r2 # Send r2 to current client

  # Cycle connection, check if we looped through whole queue
  add r1, r0, 2
  io r3, ENET_CONN_CTRL, r1
  cmpeq r3, 0
+ jup 01000
```