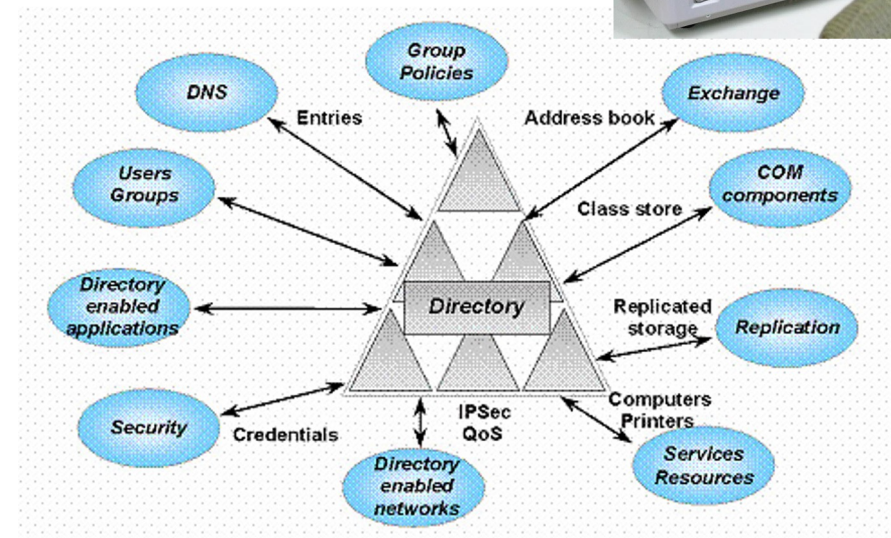


Directory Service

IN2140 Home Exam v2024

What does a directory service do?

- Directory services can be primitive or complex
- In principle, a directory server responds to queries from a user
 - The answer depends on the query
 - But also on the user and their attributes
 - Responses don't change when query and attributes are unchanged and the directory has not changed
- Different from web search, which sacrifices accuracy for scale and speed

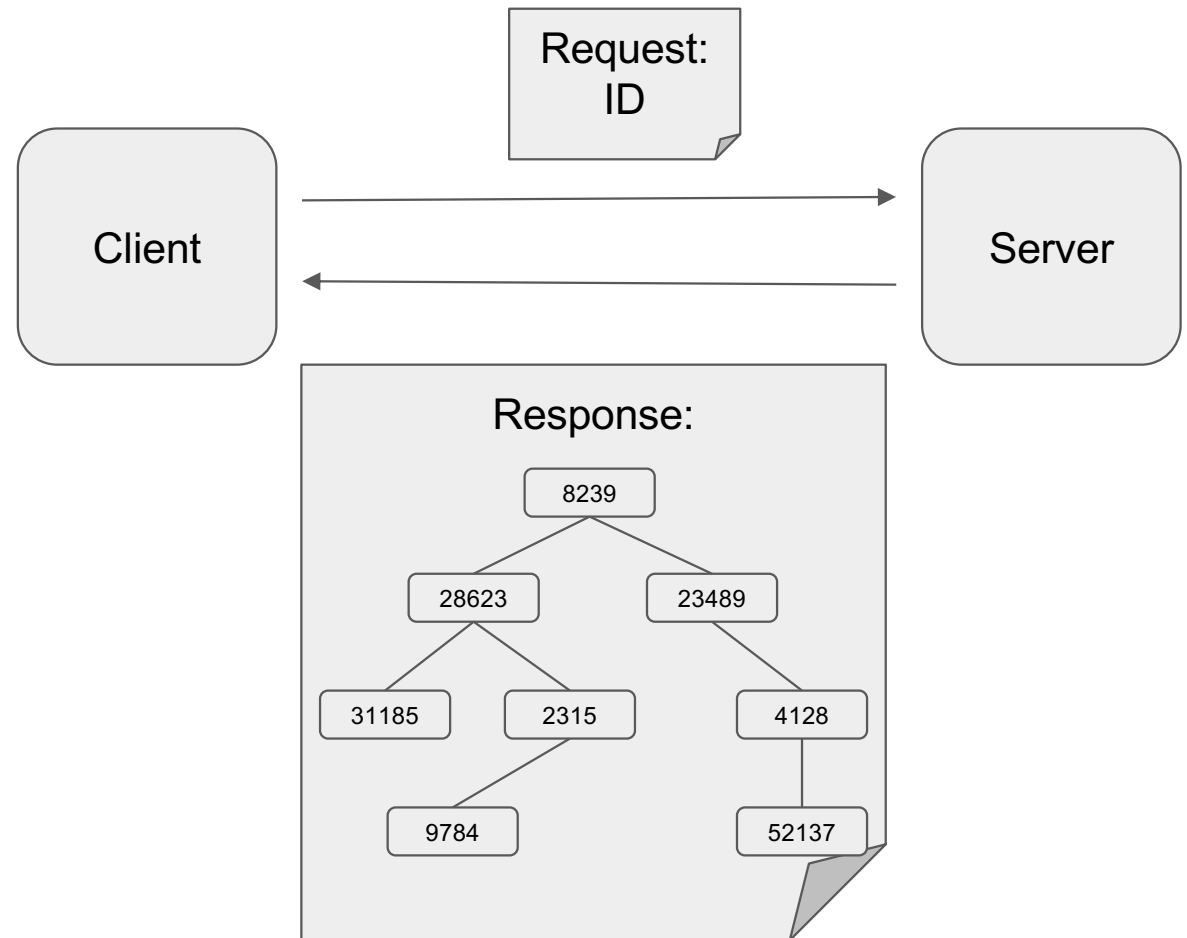


In the exam

A client asks for the directory information for an ID (ID>1000).

The server retrieves several values that are structured in a tree.

The values are simplified to some integers. In a real application these could be user permissions, files, ...

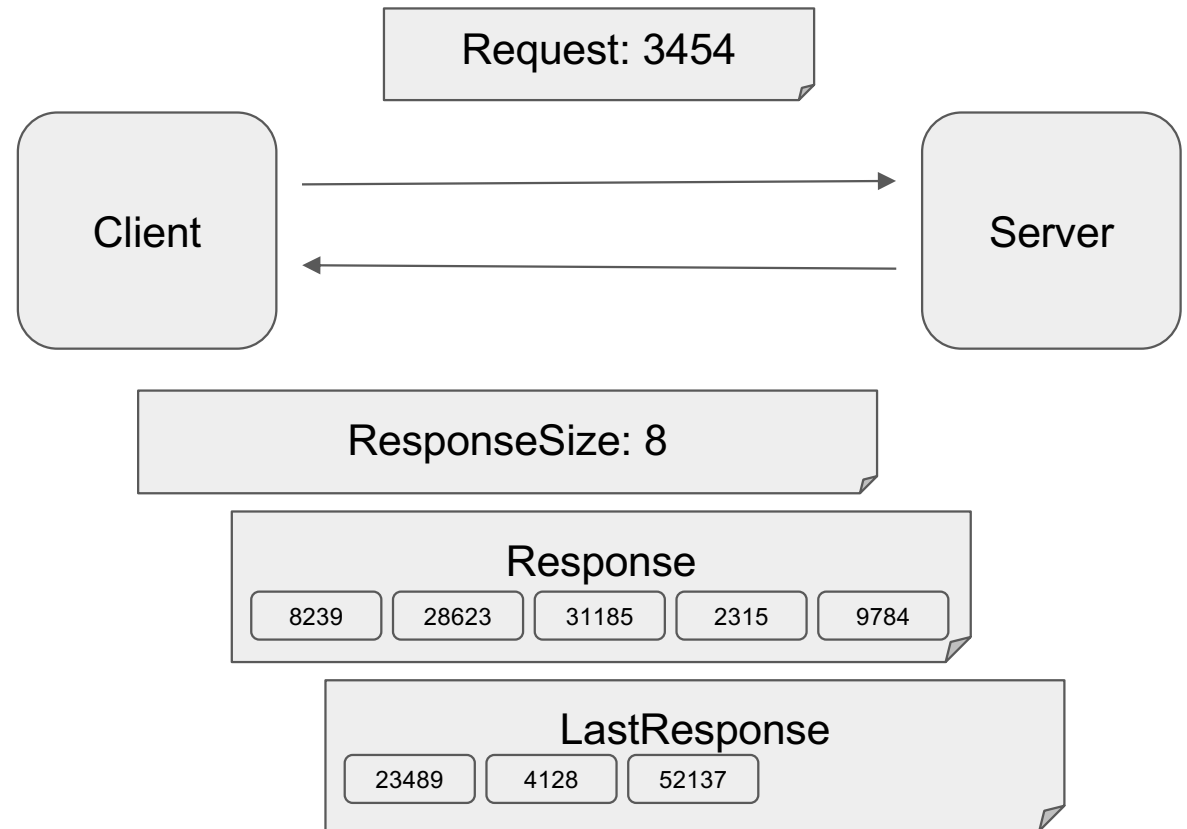


In the exam

A client asks for the directory information for an ID (ID>1000).

The server retrieves several values that are structured in a tree.

The values are simplified to some integers. In a real application these could be user permissions, files, ...

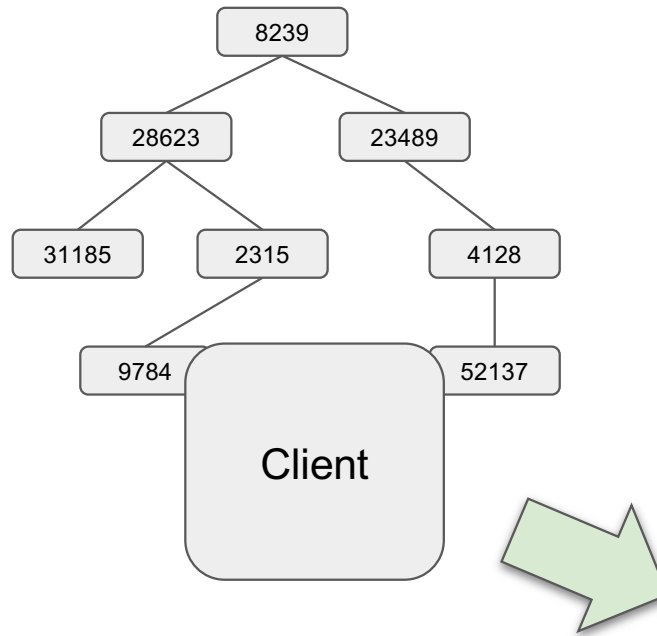


In the exam

The client prints the received response in a tree pattern to the screen.

The output shows the tree structure in a simplified form.

Note: the client and server can store the tree internally however they want.

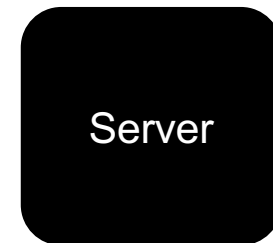
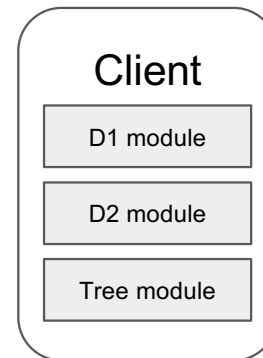


```
8239
--28623
----31185
----2315
-----9784
--23489
----4128
-----52137
```

In the exam

You implement functions in the client

- The client is a white box, you have the source.
- The client functions are in grey. They are yours, you have some precode.
- The server is a black box, you find binaries on Github.



3 steps to do this

1. Communication over UDP

- a. called “D1”
- b. with checksums and 1-bit acknowledgements

2. Request/Response protocol

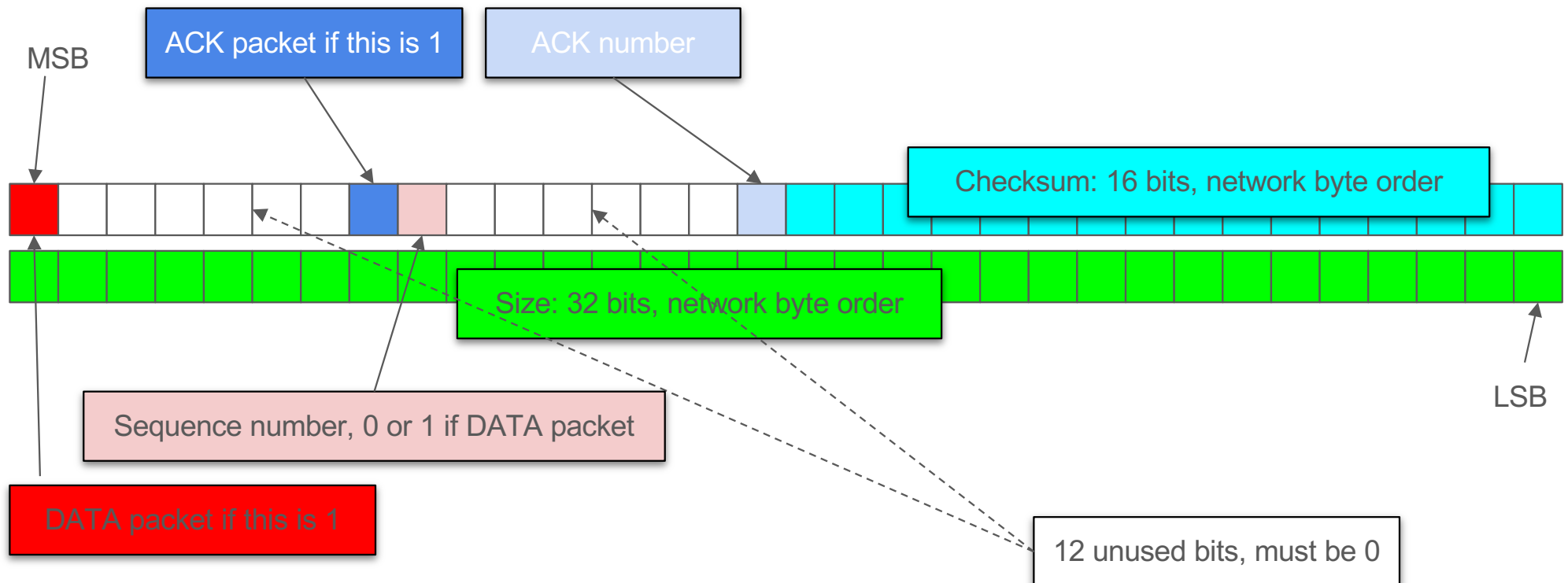
- a. called “D2”
- b. on top of UDP
- c. relies on correct packet delivery using D1

3. Build a client-side tree from the Responses

- a. Parse the payload of the D2 packets
- b. Recreate the tree on the client side (you don't have to use pointers)
- c. Print the tree on screen

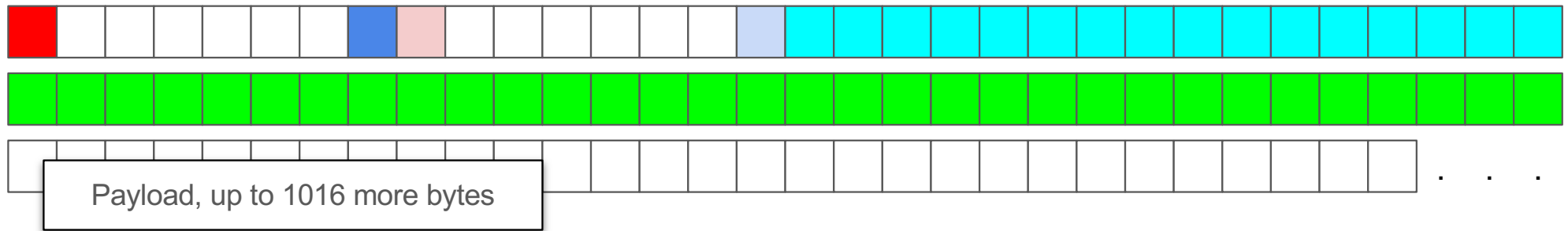
Communication over UDP

The packet header for D1: sent over the network in network byte order



Communication over UDP

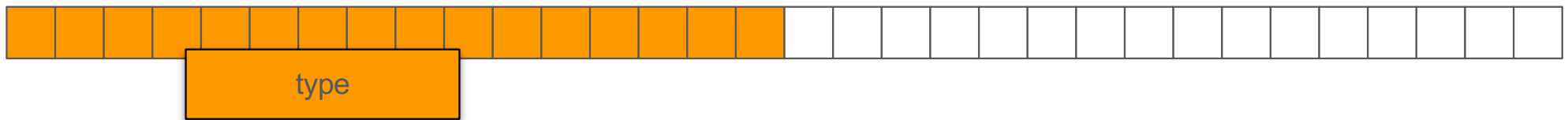
The D1 packet: payload follows D1Header, protected by checksum



```
struct D1Header
{
    uint16_t flags;
    uint16_t checksum;
    uint32_t size;
};
typedef struct D1Header D1Header;
```

D2: Request/Response protocol

The D2 packet: in the payload of the D1 packets (as usual in layering)

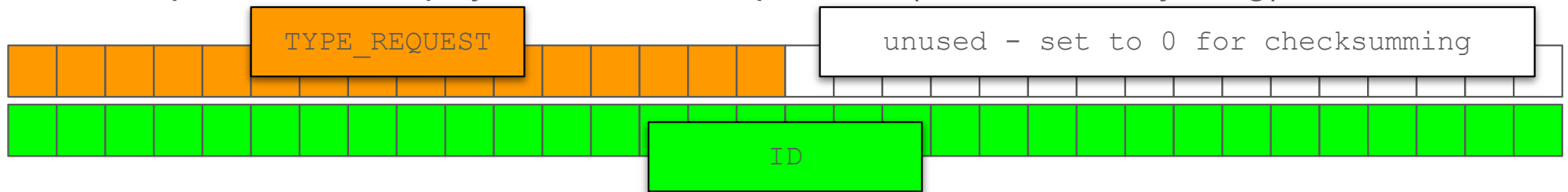


```
struct PacketHeader
{
    uint16_t type;
};
```

```
#define TYPE_REQUEST          (1 << 0) /* type is PacketRequest */
#define TYPE_RESPONSE_SIZE   (1 << 1) /* type is PacketResponseSize */
#define TYPE_RESPONSE        (1 << 2) /* type is PacketResponse */
#define TYPE_LAST_RESPONSE   (1 << 3) /* type is PacketResponse */
```

D2: Request/Response protocol

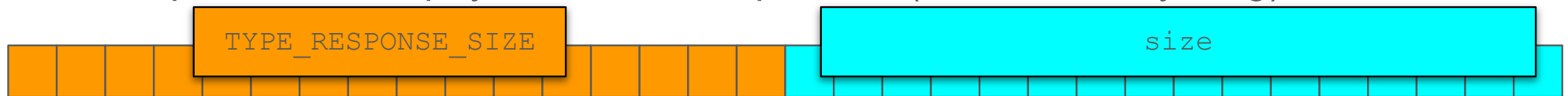
The D2 packet: in the payload of the D1 packets (as usual in layering)



```
struct PacketRequest
{
    uint16_t type;
    uint32_t id;
};
```

D2: Request/Response protocol

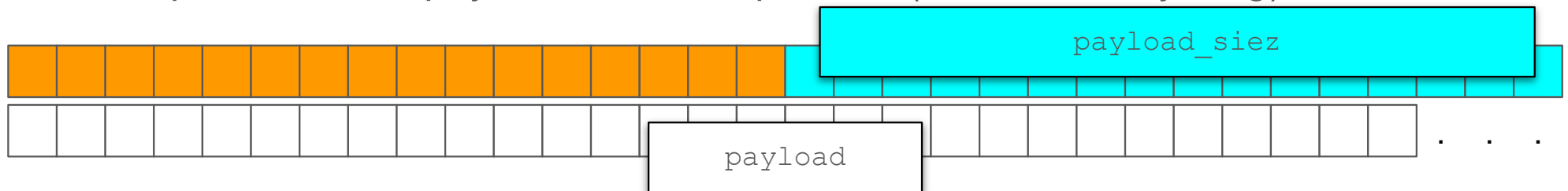
The D2 packet: in the payload of the D1 packets (as usual in layering)



```
struct PacketResponseSize
{
    uint16_t type;
    uint16_t size;
};
```

D2: Request/Response protocol

The D2 packet: in the payload of the D1 packets (as usual in layering)



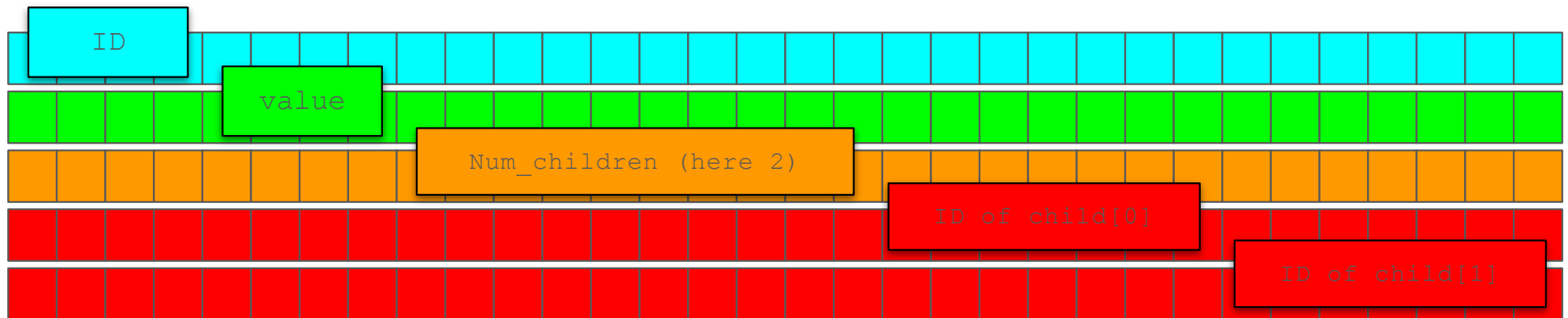
```
struct PacketResponse
{
    uint16_t type;
    uint16_t payload_size;
};
```

TYPE_RESPONSE

OR

TYPE_LAST_RESPONSE

Tree nodes: the payload of D2



```
struct NetNode
{
    uint32_t id;
    uint32_t value;
    uint32_t num_children;
    uint32_t child_id[5];
};
```

id: 0-based index created by depth-first search on the server

value: our MacGuffin

Up to 5 children
But unused children fields are not sent over the network.

The servers

Your client code must talk to our servers

Find binaries at <https://github.uio.no/IN2140v2/in2140-v24-he>

- **d1_dump <port>**
 - Send one of your D1 packets to this server. The server checks whether your D1 header is as expected.
- **d1_server <port>**
 - The d1_test_client with your implementation of d1_udp.c can connect to this server and send a few ping-pong messages.
- **d2_server <port>**
 - The d2_test_client with your implementations of d1_udp.c and d2_lookup.c can connect to this server. The client sends a Request with an ID and the server answers with several Response packets with tree nodes.

The servers

Your client code must talk to our servers

Find binaries at <https://github.uio.no/IN2140v2/in2140-v24-he>

- Currently available for
 - Linux Redhat 8.9 on Intel (static binary)
 - this works on IFI's computers, login.ifi and in machines in Sed
 - Linux Ubuntu 22.04 on Intel (static binary)
 - this is popular for home use (and may work with WSL2)
 - MacOS 14.4 Sonoma on Intel (dynamic binary)
 - only Intel; no access to an ARM Mac