Chair of Distributed Systems and Security
School of Computation, Information and Technology
Technical University of Munich

Computer Networking and IT Security (INHN0012)

Tutorial 8

Problem 1 IPv6 and Supernetting

TUMexam AG has now been assigned the IPv6 address ranges `2001:0db8:0001:000d:0000:0000:0000:0000/64` (*NET1*) and `2001:0db8:0001:000e:0000:0000:0000:0000/64` (*NET2*).

a)* Specify the IPv6 address contained in *NET1* `2001:0db8:0001:000d:0000:00f0:0000:0000` in its compact notation.

- leading zeros are omitted: `2001:db8:1:d:0:f0:0:0`
- the largest consecutive block of at least 2 „zero “ blocks can be abbreviated by `::` `2001:db8:1:d:0:f0::`

b)* How many addresses does each prefix contain?

$2^{128-64} = 18\,446\,744\,073\,709\,551\,616$

c) How many times can the entire IPv4 address range (`0.0.0.0/0`) be mapped into *NET1*?

$2^{(128-64)-32} = 2^{32} = 4\,294\,967\,296$

d)* What conditions must be met for 2 subnets to be aggregated?

- same size, which implies same prefix length n
- adjacent (the last address in the first network must be followed directly by the next network)
- There must exist a valid prefix mask with length $n-1$, i.e. h. the two nets must differ only exactly in the last bit of their prefix.

e)* Can the two subnets *NET1* and *NET2* be aggregated into one `/63` subnet?

Although the nets are of the same size and are adjacent, they cannot be aggregated because they are not in the same `/63` prefix.For bits 61 to 64: $d_{16} = 1101_2$, $e_{16} = 1110_2$.
`2001:db8:1:c::/62` would include the two networks, but in addition would also include

- `2001:db8:1:c::/64` and
- `2001:db8:1:f::/64`.

Problem 2 ARP and IP fragmenting

Figure 2.1 shows an arrangement of network components with their IP and MAC addresses. The two computers PC1 and PC2 use the respective local router as default gateway. PC1 sends an IP packet with 1000 B payload data to PC2. The MTU on the WAN link between R1 and R2 is 580 B. Within the local networks the usual Ethernet MTU of 1500 B applies.

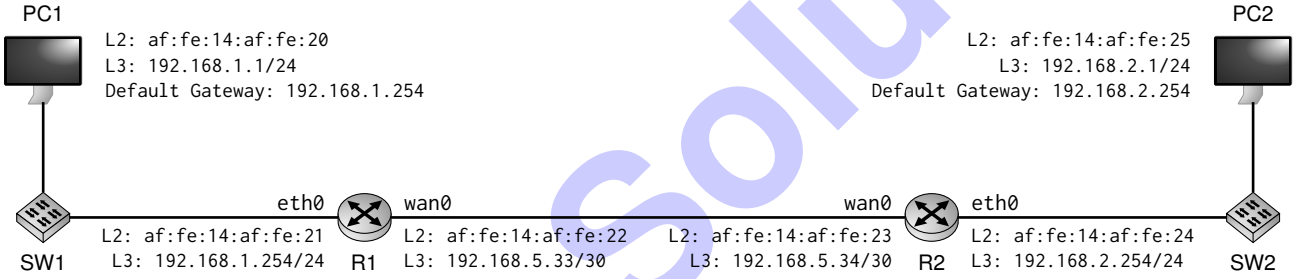


Figure 2.1: Network topology

In the following, the transfer of the packet with all necessary intermediate steps shall be traced. Assume initially that the ARP caches of all network components involved have been flushed.

a)* To what extent do the two switches SW1 and SW2 have an effect in this example?

The switches have no influence on the exchanged messages.Switches are usually transparent to the connected hosts.In particular, switches do not change the sender or recipient address.

b)* Into how many fragments must R1 break down the packet from PC1?

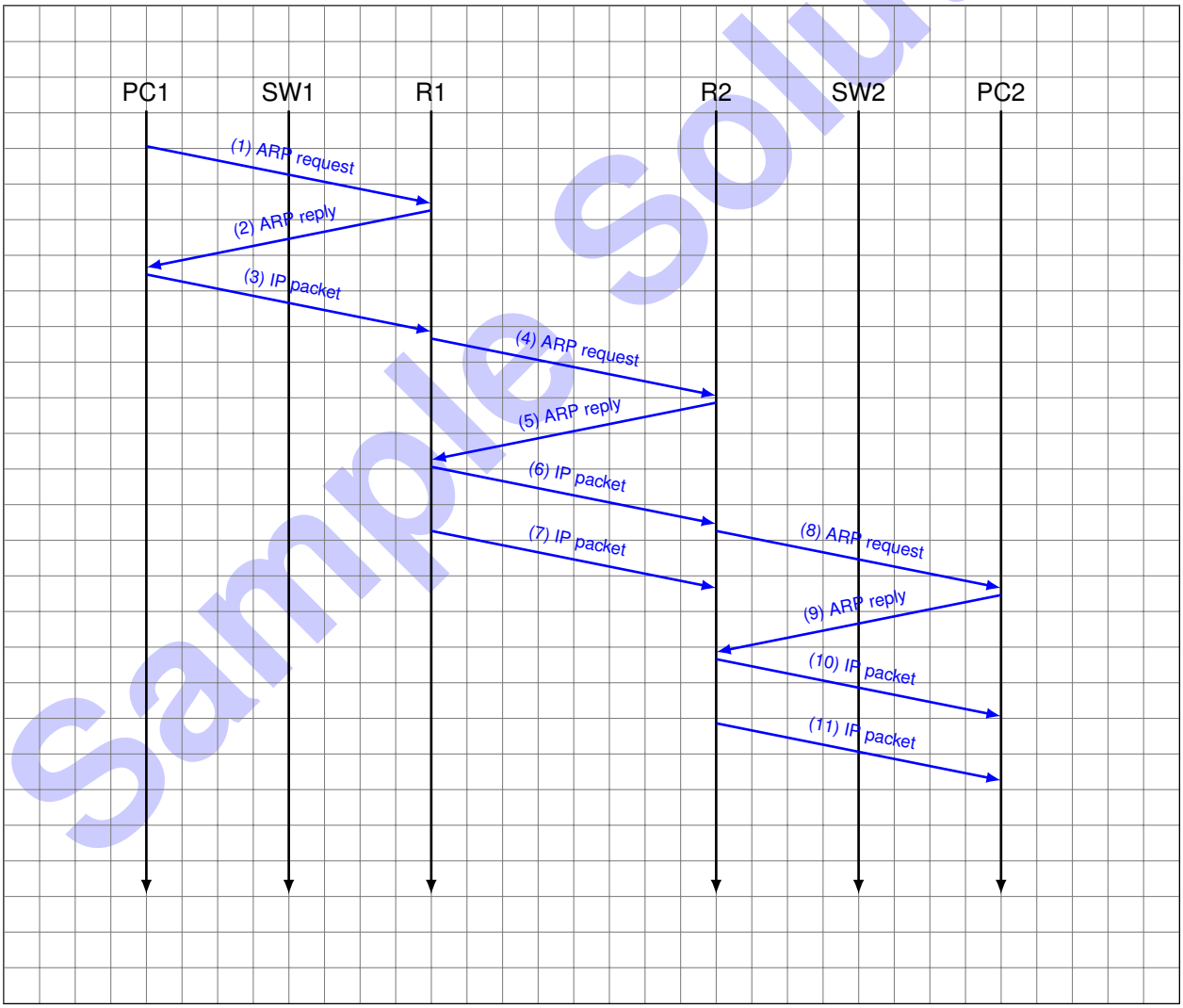
The MTU (Maximum Transmission Unit) is the maximum size of a packet on layer 3 incl. header. It is therefore exactly the same as the maximum size of the payload on layer 2. Knowing that an IP header is 20 B long (exception when using options), we get:

$$N = \left\lceil \frac{1000\,\mathrm{B}}{580\,\mathrm{B} - 20\,\mathrm{B}} \right\rceil = 2$$

c)* At what point in the network are the fragments reassembled?

Only the receiver, here PC2, reassembles the fragments.In fact, no other node can perform the reassembly, since the fragments represent individual and independent packets.
In particular, this means that they are routed independently and may therefore take different paths to the destination – of course, this is not seen from the simple example in Figure 2.1, where there is only one path between PC1 and PC2.

d) Sketch a simple path-time diagram that considers **all frames** that need to be transferred on each link. **Name the type of frames replaced and number the frames (1,2,3,...).** (The diagram does not need to be to scale. Serialization times and propagation delays are to be ignored.)



At the end of this exercise sheet you will find preprinted forms for Ethernet header, ARP packets (header and payload) and IP header (more than needed). It is not necessary to fill in the header in binary. Just be sure to clearly mark the number base, e.g., `0x10` for hexadecimal or $63_{(10)}$ for decimal.

e) For each of the first three frames from subtask d), fill in an Ethernet header and the appropriate payload (ARP packet or IP header with indicated payload). Label the dashed box next to each header/packet with the frame number assigned in subtask d).

f) Fill in an Ethernet and IP header for each of the remaining frames that transport an IP payload. Label the dashed box next to each header with the frame number assigned in subtask d).

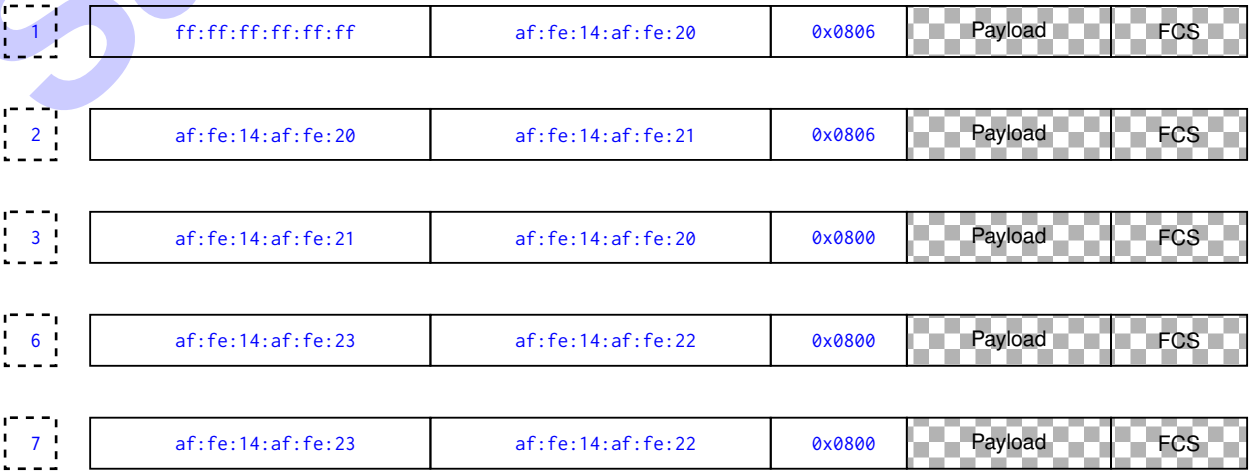
g)* Assume that PC1 and PC2 communicate via IPv6:

1. What impact would this have on switches SW1 and SW2?
2. In this case, would routers R1 and R2 also have to be IPv6-capable?
3. Where would the fragmentation of packets take place?

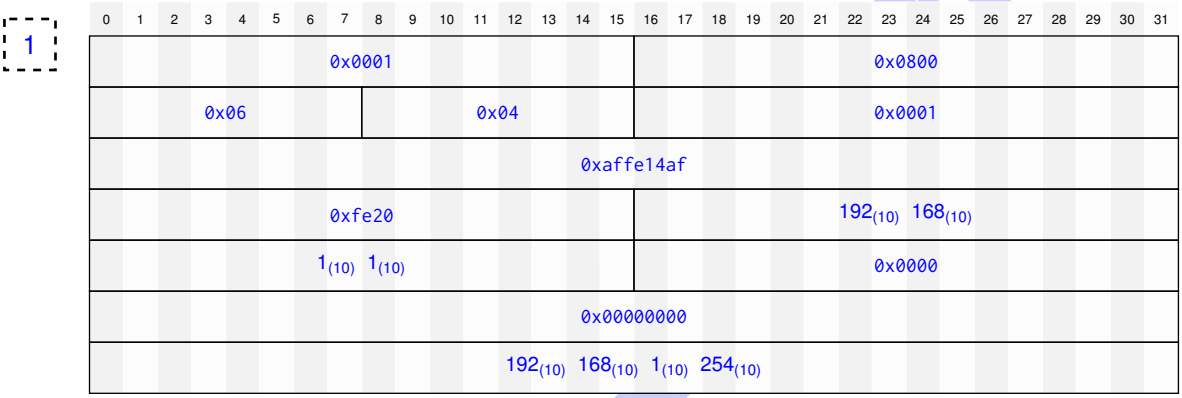
1. In the given case, none at all: switches work only with MAC addresses, which would not change (except for multicast, if applicable).
2. Yes, at least on the local interfaces `eth0`, because IPv6 and IPv4 are not compatible. Transporting IPv6 over IPv4 using GRE (General Routing Encapsulation) is theoretically possible, but not very useful or generally possible due to the non-injective mapping of IPv4 to IPv6.
3. Fragmentation would now take place directly at PC1, since IPv6 routers do not fragment at all.

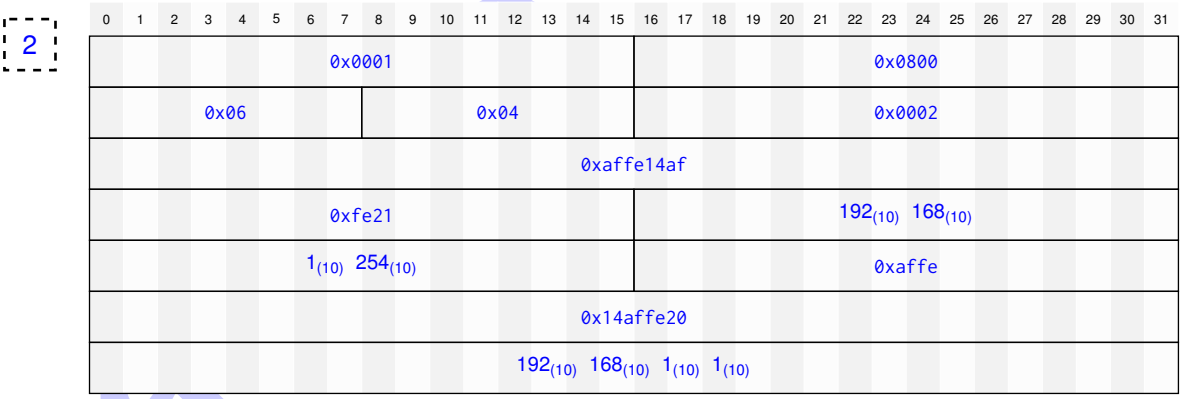
Preprints for protocol headers:

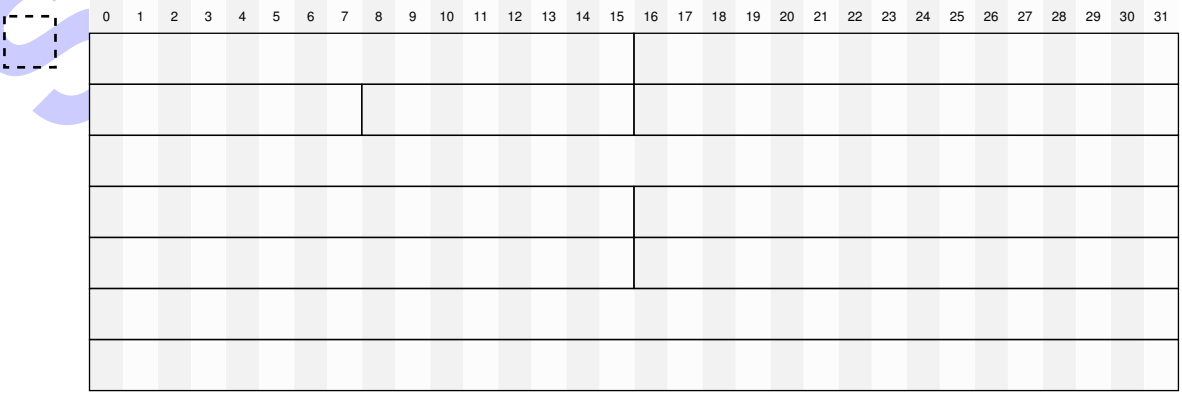
Ethernet frames



ARP packets

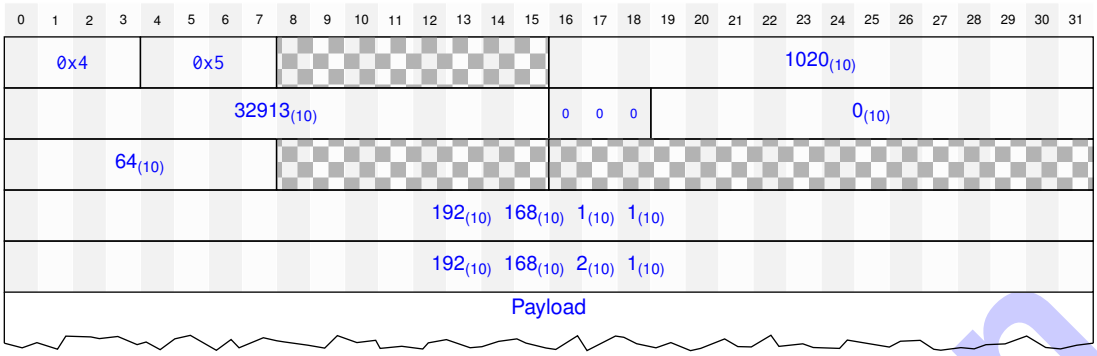




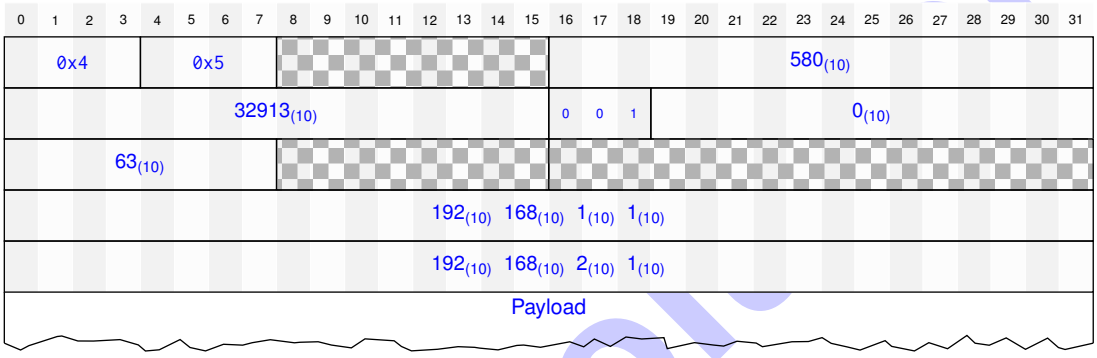


IP packets

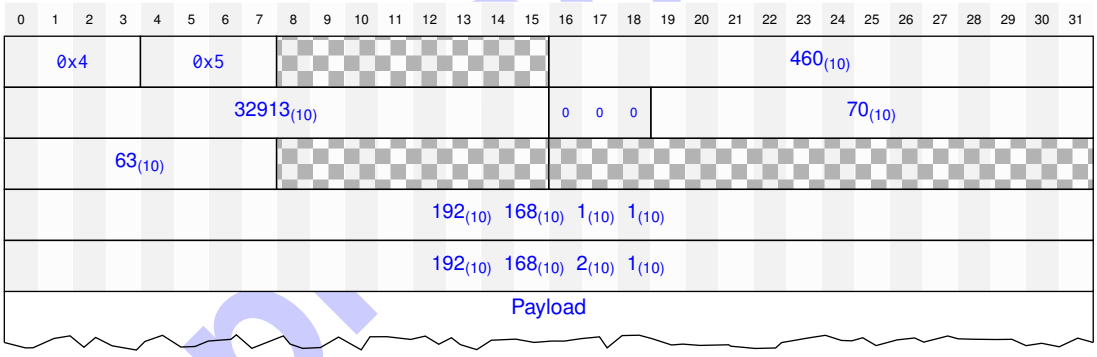
3


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0x4
0x5
1020(10)
32913(10)
0 0 0
0(10)
64(10)
192(10) 168(10) 1(10) 1(10)
192(10) 168(10) 2(10) 1(10)
Payload

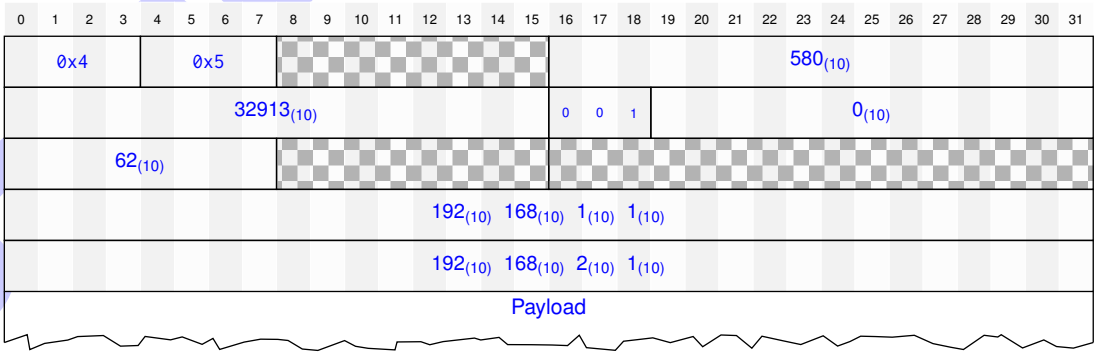
6


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0x4
0x5
580(10)
32913(10)
0 0 1
0(10)
63(10)
192(10) 168(10) 1(10) 1(10)
192(10) 168(10) 2(10) 1(10)
Payload

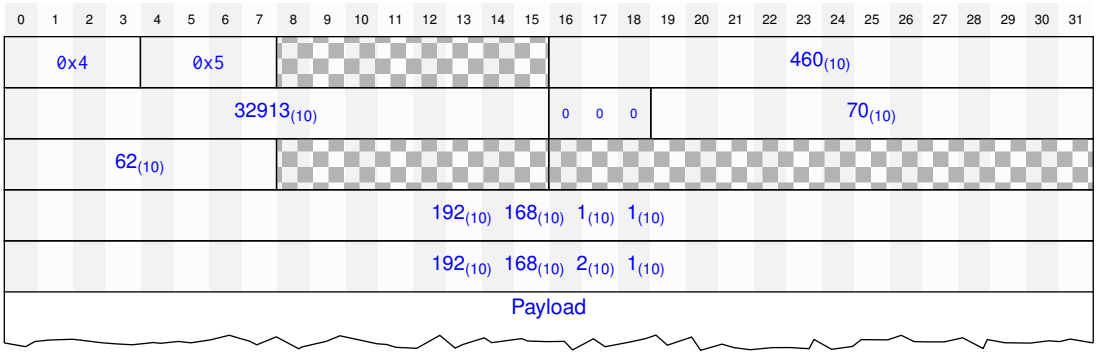
7

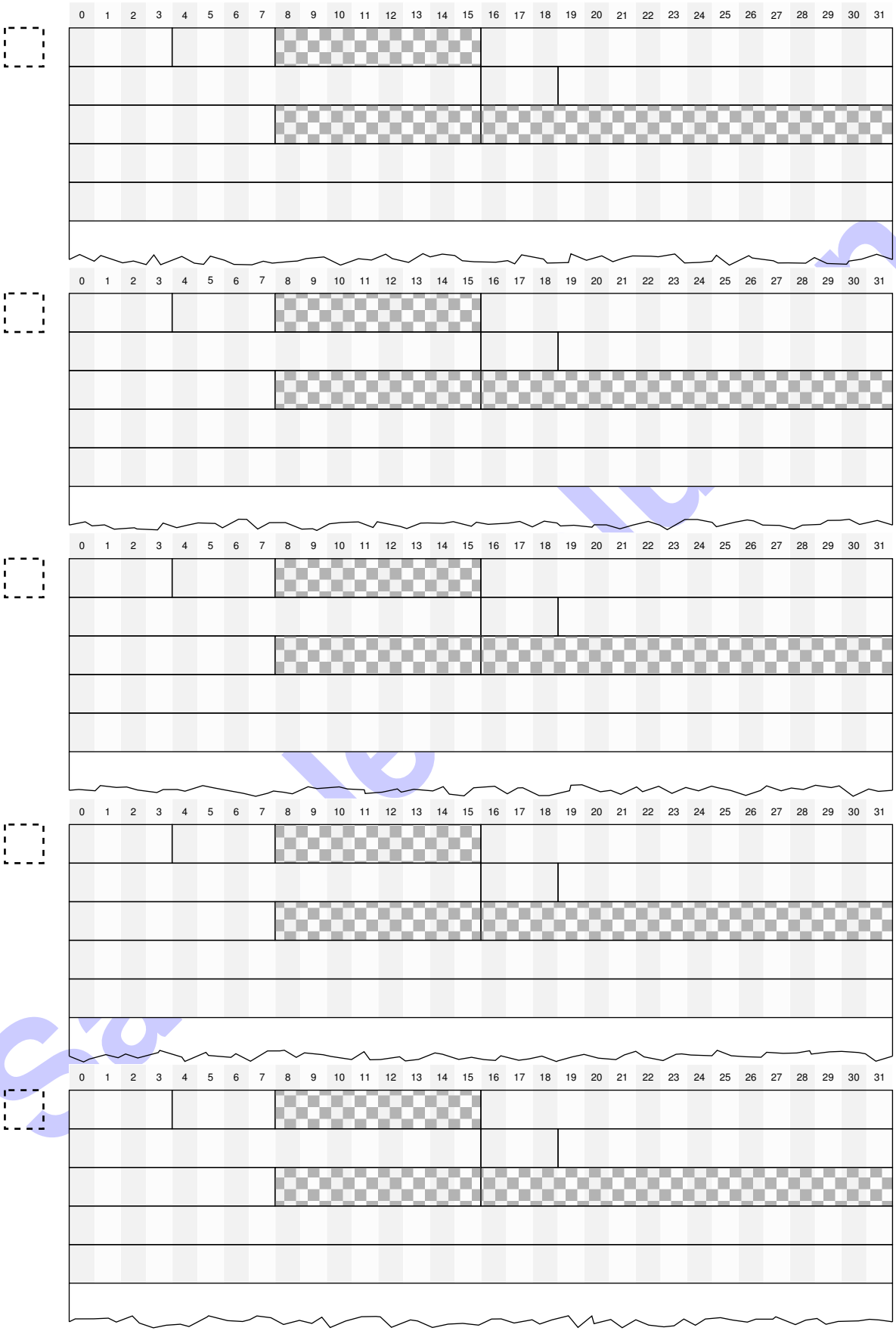

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0x4
0x5
460(10)
32913(10)
0 0 0
70(10)
63(10)
192(10) 168(10) 1(10) 1(10)
192(10) 168(10) 2(10) 1(10)
Payload

10


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0x4
0x5
580(10)
32913(10)
0 0 1
0(10)
62(10)
192(10) 168(10) 1(10) 1(10)
192(10) 168(10) 2(10) 1(10)
Payload

11


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0x4
0x5
460(10)
32913(10)
0 0 0
70(10)
62(10)
192(10) 168(10) 1(10) 1(10)
192(10) 168(10) 2(10) 1(10)
Payload


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31