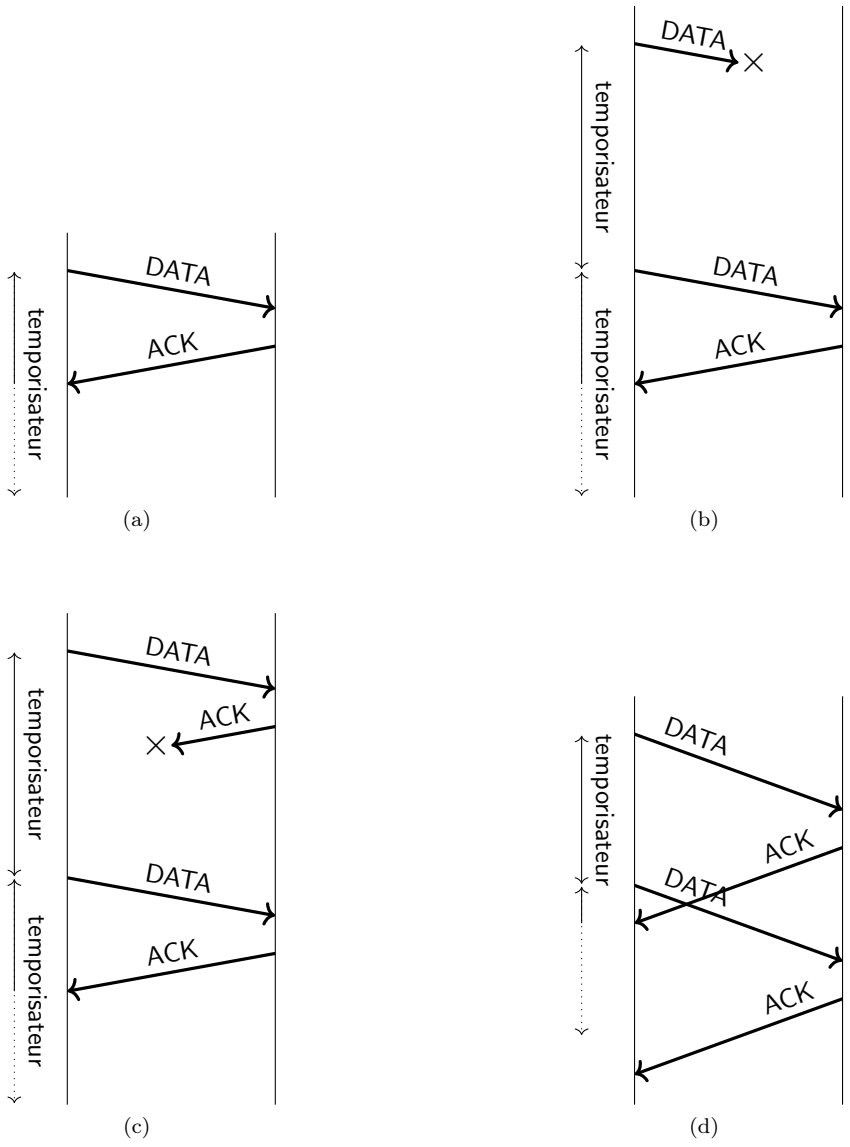


FIGURE 1 – Stop and Wait (incorrect !)

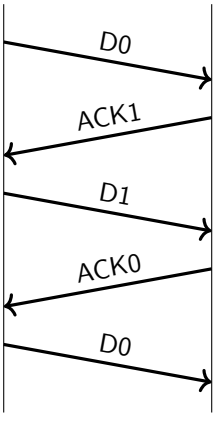


FIGURE 2 – Stop and Wait avec numéros de séquence sur 1 bit

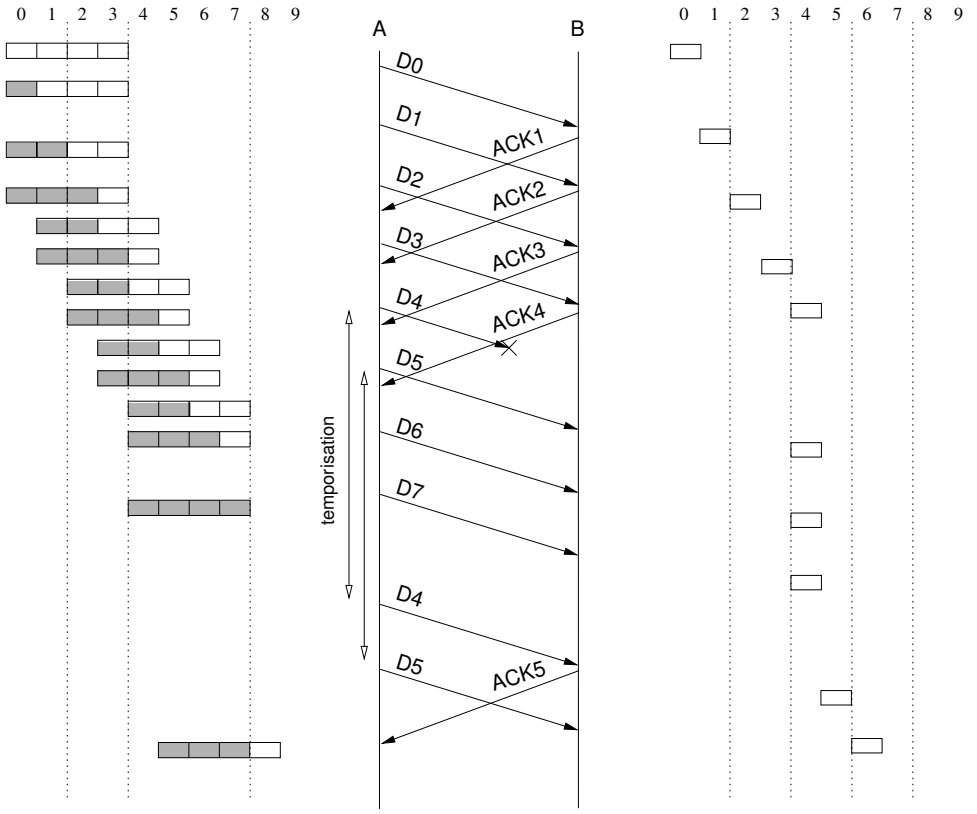


FIGURE 3 – Transmission en fenêtre glissante. TFE = 4 *(go-back-n)*

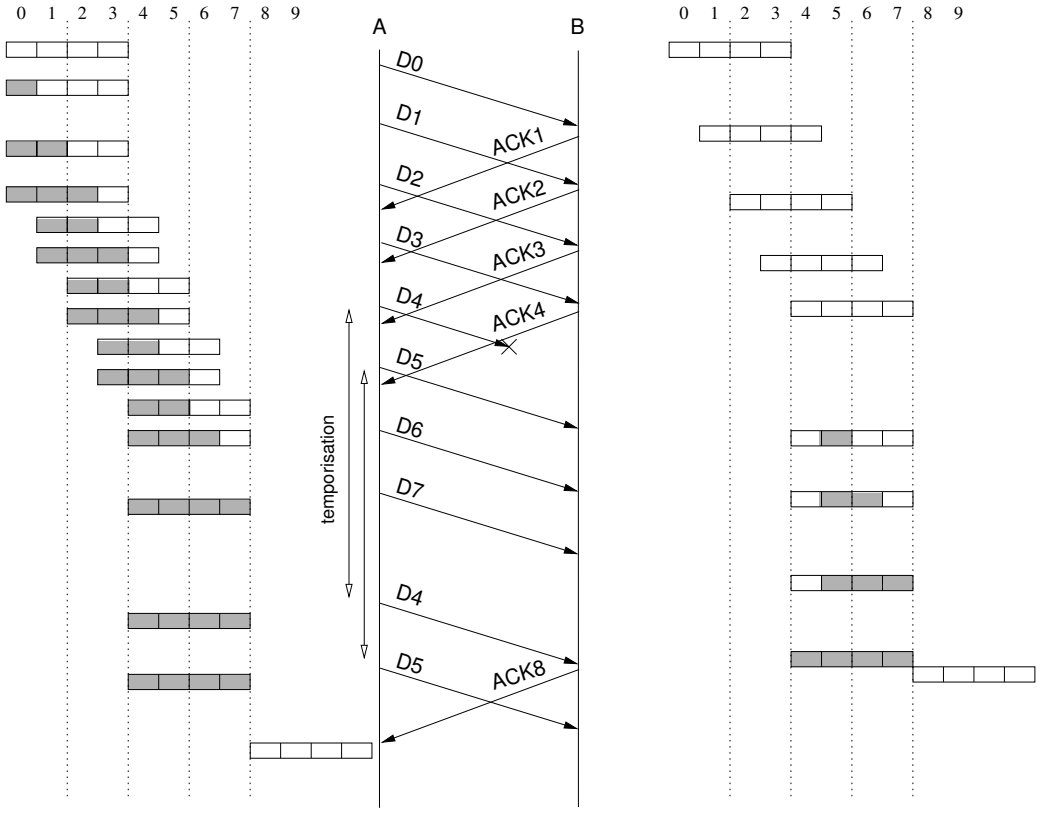


FIGURE 4 – Transmission en fenêtre glissante. TFE = 4, TFR = 4 *(selective repeat)*

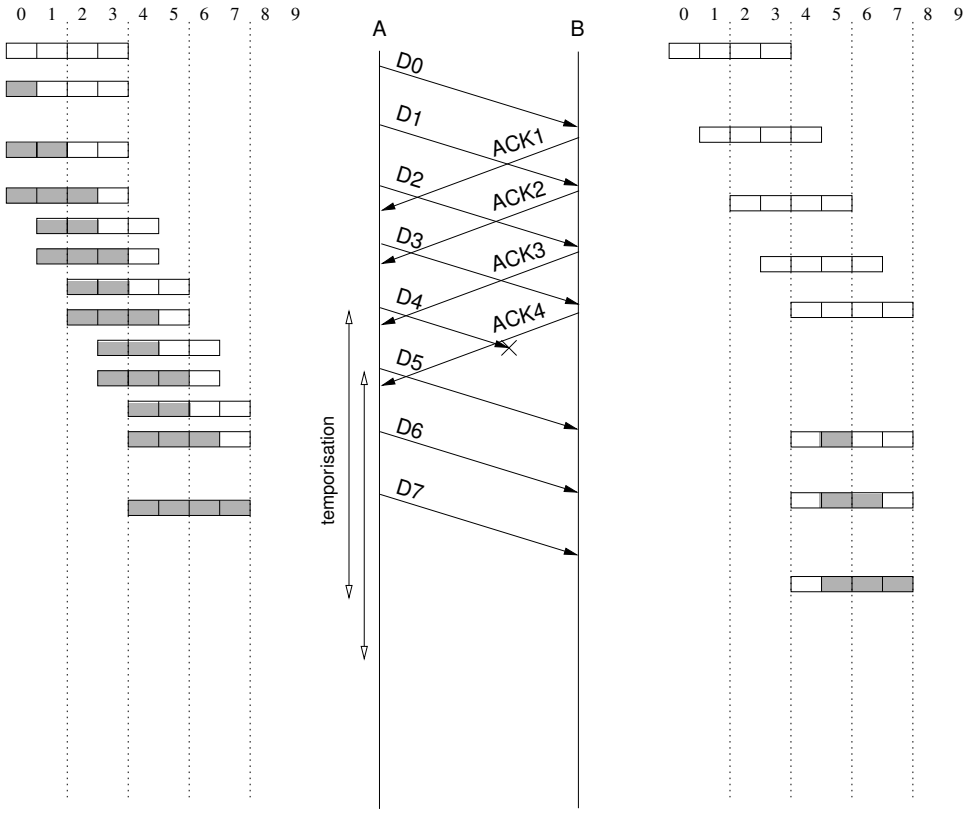


FIGURE 5 – TFE = 4, TFR = 4 et reprise rapide (à compléter)

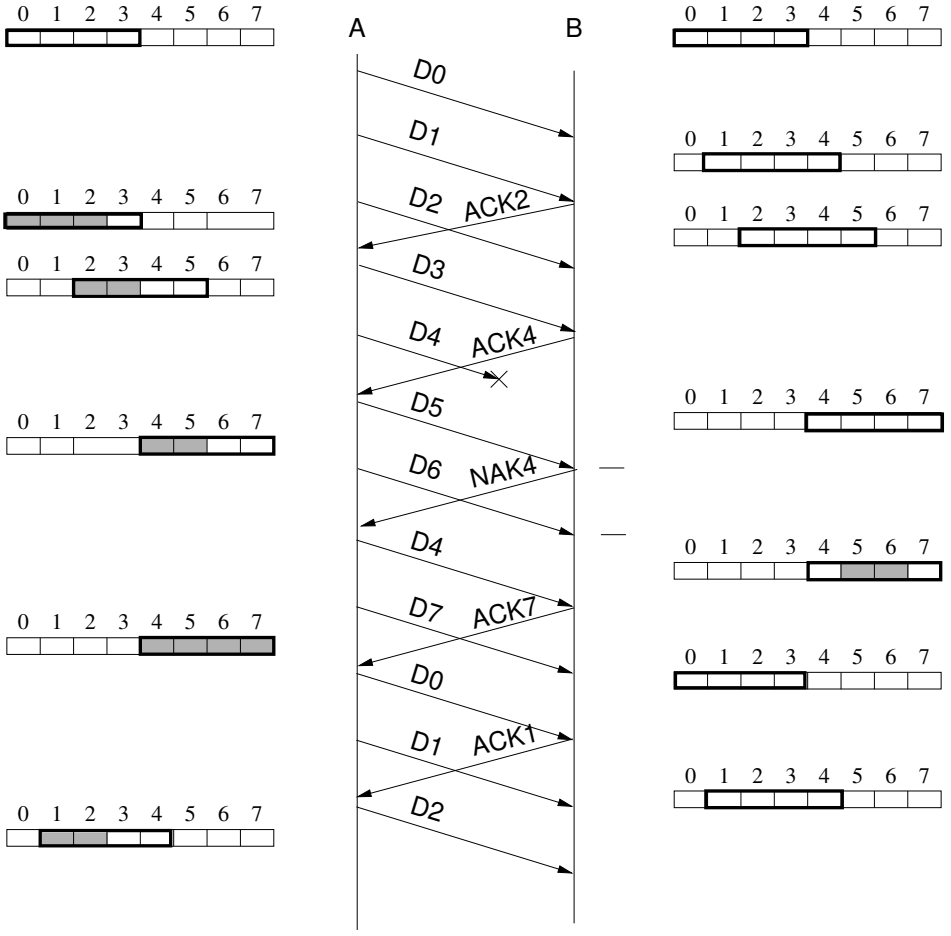


FIGURE 6 – numéros de séquence finis, NSEQ=8, TFE = 4, TFR = 4 (incomplet)

```
#define NSEQ ...     /* NSEQ sequence numbers, in [0, NSEQ[ */

#define SWS ...      /* send window size */
#define RWS ...      /* receive window size */

typedef enum {data, ack} frame_kind; /* two kinds of frame */

typedef struct {     /* frames are transported in this layer */
  frame_kind kind;   /* what kind of a frame is it? */
  uint_t seq;        /* sequence number */
  uint_t ack;        /* acknowledgement number */
  packet data;       /* the network layer packet */
} frame;

static void send_data_frame(uint_t seq, uint_t ack, packet *buf)
{
    frame s;

    s.kind = data;
    s.data = buf[seq];
    s.seq  = seq;
    s.ack  = ack;

    to_physical_layer(&s); /* transmit the frame */
    start_timer(seq);
    stop_ack_timer();      /* since we just sent a piggybacked ack */
}

static void send_ack_frame(uint_t seq)
{
    frame s;

    s.kind = ack;
    s.ack  = seq;

    to_physical_layer(&s);    /* transmit the frame */
    stop_ack_timer();         /* since we just sent one */
}
```

FIGURE 7 – Mise en œuvre du protocole à fenêtre glissante (1/2)

```
typedef enum {frame_arrival, cksum_err, timeout, network_layer_ready, ack_timeout} event_type;

#define inc_seq(k)  k = (k + 1) % NSEQ

void swp1()
{
    uint_t sw1 = 0;           /* first non acked frame */
    uint_t sw2 = 0;           /* next frame to send */
    uint_t nbuffered = 0;     /* how many sent frames waiting for ack */
    packet snd_buf[NSEQ];     /* sending buffers */

    uint_t rw1 = 0;           /* expected frame  */
    packet rcv_buf[NSEQ];     /* receive buffers */
    bool   arrived[NSEQ];     /* which are used  */

    int i;
    frame r;
    event_type event;

    enable_network_layer();
    for (i = 0; i < NSEQ; i++) arrived[i] = false;

    while (true) {
        wait_for_event(&event);
        switch(event) {

        case network_layer_ready:       /* accept, save, and transmit a new frame */
            nbuffered++;
            from_network_layer(&snd_buf[sw2]);
            send_data_frame(sw2, rw1, snd_buf);
            inc_seq(sw2);
            break;

        case frame_arrival:             /* a data or control frame has arrived */
            from_physical_layer(&r);
            if ((r.kind == data) &&     /* data frame */
                 in_rcv_window(r.seq, rw1, RWS) && (arrived[r.seq] == false)) {  /* this is a new frame */
                   arrived[r.seq] = true;
                   rcv_buf[r.seq] = r.data;

                   while (arrived[rw1] == true) { /* Pass packets and slide receiver's window */
                       to_network_layer(&rcv_buf[rw1]);
                       arrived[rw1] = false;
                       inc_seq(rw1);
                       start_ack_timer();    /* to see if a separate ack is needed */
                   }
            }

            while (is_new_ack(r.ack, sw1, sw2)) {      /* handle ACK */
                nbuffered = nbuffered - 1;
                stop_timer(sw1);
                inc_seq(sw1);        /* slide sender's window */
            }
            break;

        case cksum_err: break;          /* damaged frame */

        case timeout:                   /* retransmission needed */
            send_data_frame(get_timedout_seqnr(), rw1, snd_buf); break;

        case ack_timeout:               /* ack timer expired; send ack */
            send_ack_frame(rw1); break;
        }

        if (nbuffered < SWS) enable_network_layer();    /* check whether sending window is full */
        else disable_network_layer();
    }
}
```

FIGURE 8 – Mise en œuvre du protocole à fenêtre glissante (2/2)

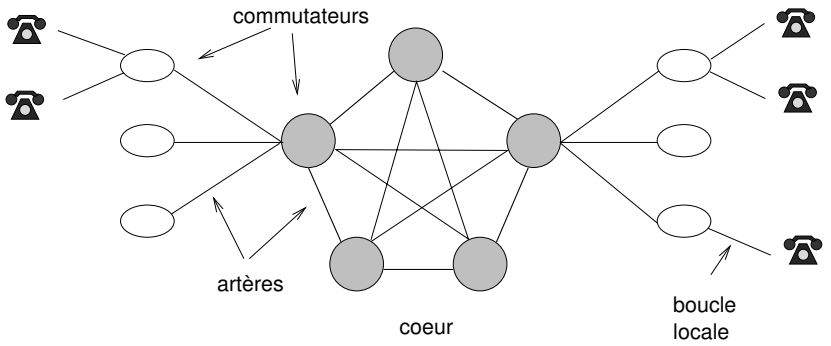


FIGURE 9 – Structure du réseau téléphonique (simplifiée à 3 niveaux)
chiffres AT&T : 10 × 67 × 230 × 1300 × 19000 × 200M

canaux	largeur (Hz)	spectre (Hz)	nom AT&T	nom ITU-T
1	3100	0,3 – 3,4 k		
12	48 k	60 – 108 k	group	group
60	240 k	312 – 552 k	supergroup	supergroup
300	1,232 M	812 – 2044 k		mastergroup
600	2,52 M	564 –3084 k	mastergroup	
900	3,872 M	8,516 – 12,388 M		supermaster group
3600	16,984 M	0,564 – 17,548 M	jumbogroup	
10800	57,442 M	3,124 – 60,566 M	jumbogroup multiplex	

TABLE 1 – Hiérarchies de multiplexage FDM AT&T et ITU-T

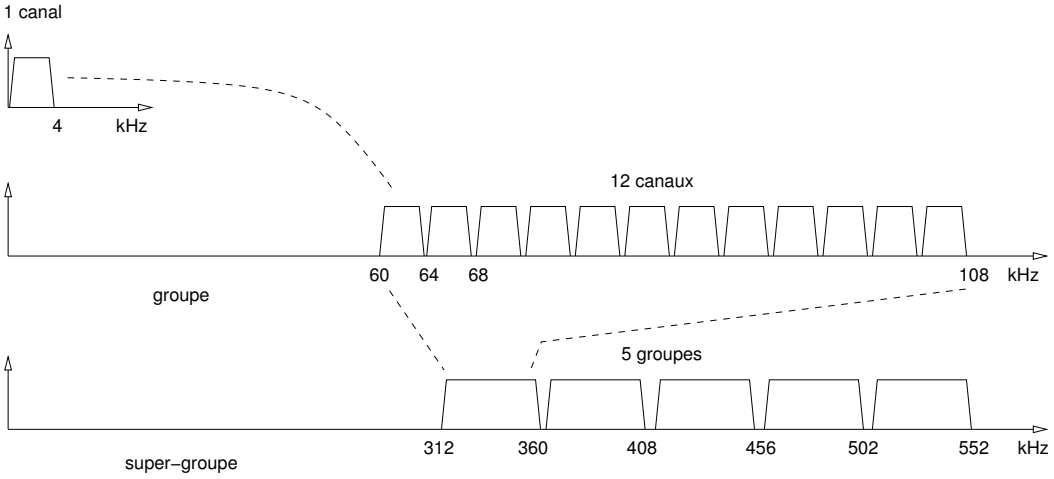


FIGURE 10 – Spectre d'un groupe et d'un super-groupe

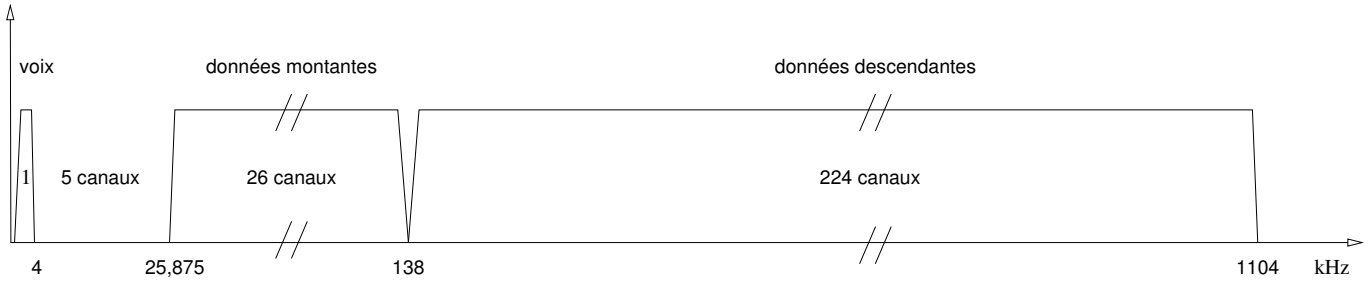
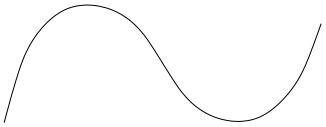
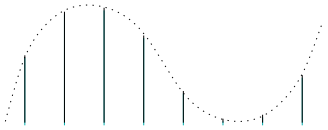


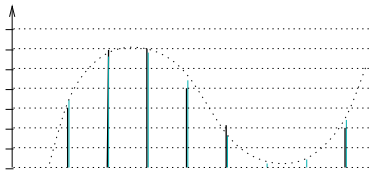
FIGURE 11 – Bandes de fréquence d'un accès ADSL



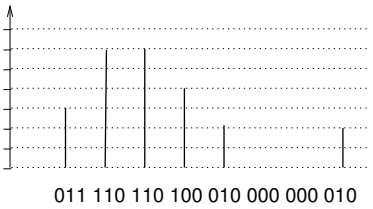
(a) signal analogique



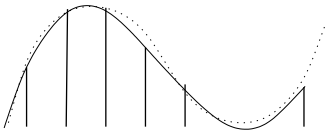
(b) échantillonage



(c) quantification (ici sur 8 niveaux)



(d) codage



(e) reconstruction

FIGURE 12 – Codage PCM

nom	format trame	canaux	débit (Mb/s)
T1	DS-1	24	1,544
T1C	DS-1C	48	3,152
T2	DS-2	96	6,312
T3	DS-3	672	44,736
T4	DS-4	4032	274,176
T5	DS-5	5760	400,352

TABLE 2 – Hiérarchie de multiplexage TDM AT&T

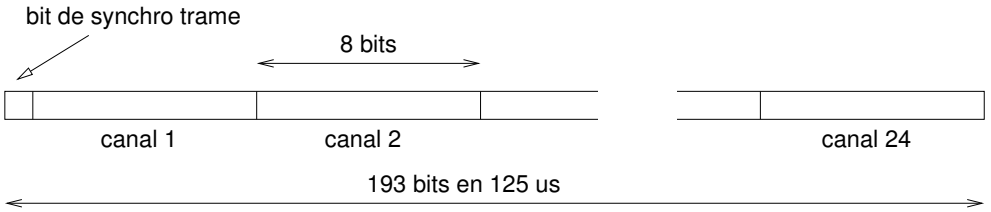


FIGURE 13 – Trame DS-1

Nom	canaux	données	débit (Mb/s)
E0	1	1	64 kb/s
E1	32	30	2,048
E2	128	120	8,448
E3	512	480	34,368
E4	2048	1920	139,264
E5	8192	7680	565,148

TABLE 3 – Hiérarchie de multiplexage TDM ITU

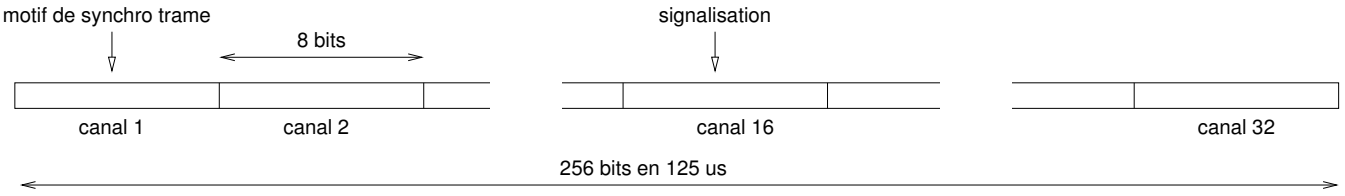


FIGURE 14 – Trame E1

SONET Optical Carrier Level	SONET Frame Format	SDH level and Frame Format	Payload rate (Mbit/s)	Line rate (Mbit/s)
OC-1	STS-1	STM-0	50.112	51.840
OC-3	STS-3	STM-1	150.336	155.520
OC-12	STS-12	STM-4	601.344	622.080
OC-24	STS-24		1 202.688	1 244.160
OC-48	STS-48	STM-16	2 405.376	2 488.320
OC-192	STS-192	STM-64	9 621.504	9 953.280
OC-768	STS-768	STM-256	38 486.016	39 813.120
OC-3072	STS-3072	STM-1024	153 944.064	159 252.240

TABLE 4 – Hiérarchie de multiplexage TDM SONET/SDH

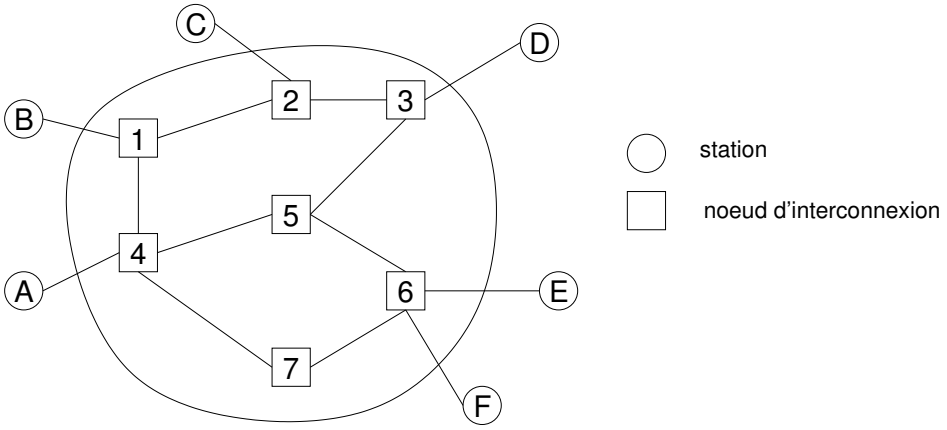


FIGURE 15 – Réseau de transit

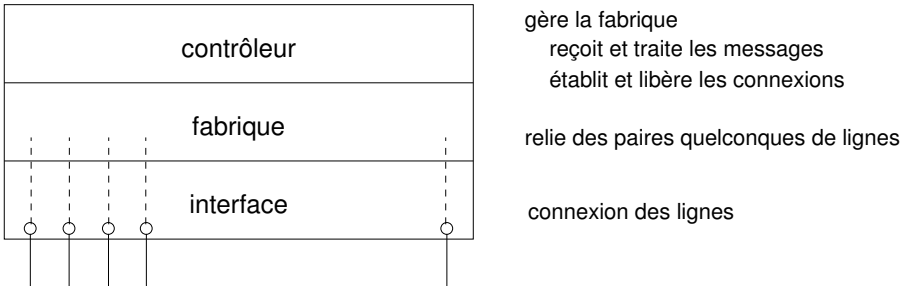
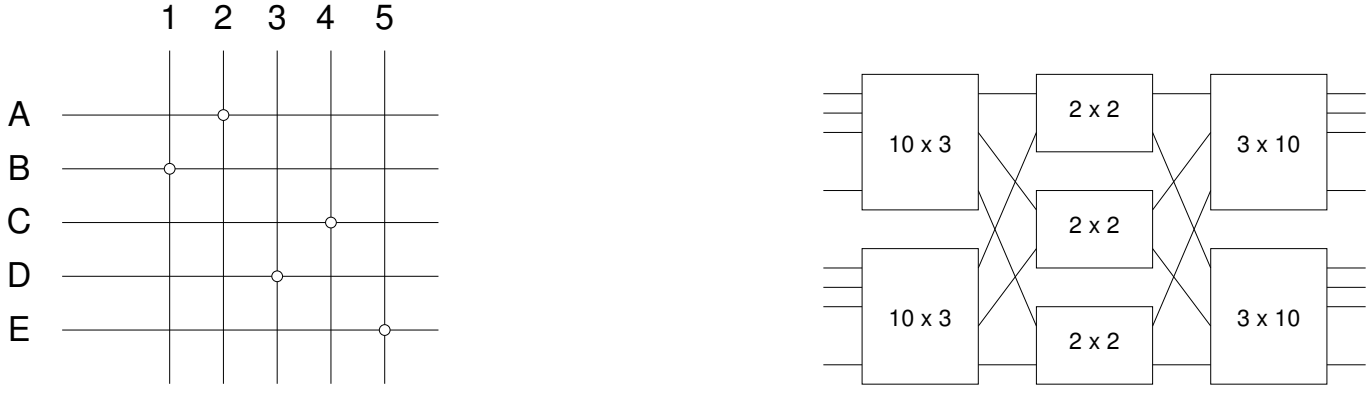


FIGURE 16 – Éléments d'un commutateur



(a) Crossbar simple 5x5

(b) Plusieurs niveaux 20x20

FIGURE 17 – Fabrique de commutateur spatial

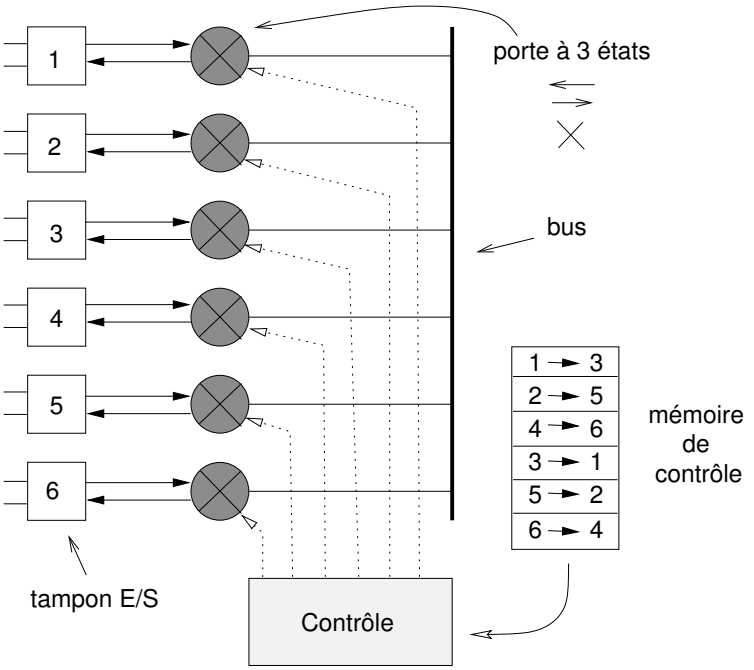


FIGURE 18 – Fabrique de commutateur temporel

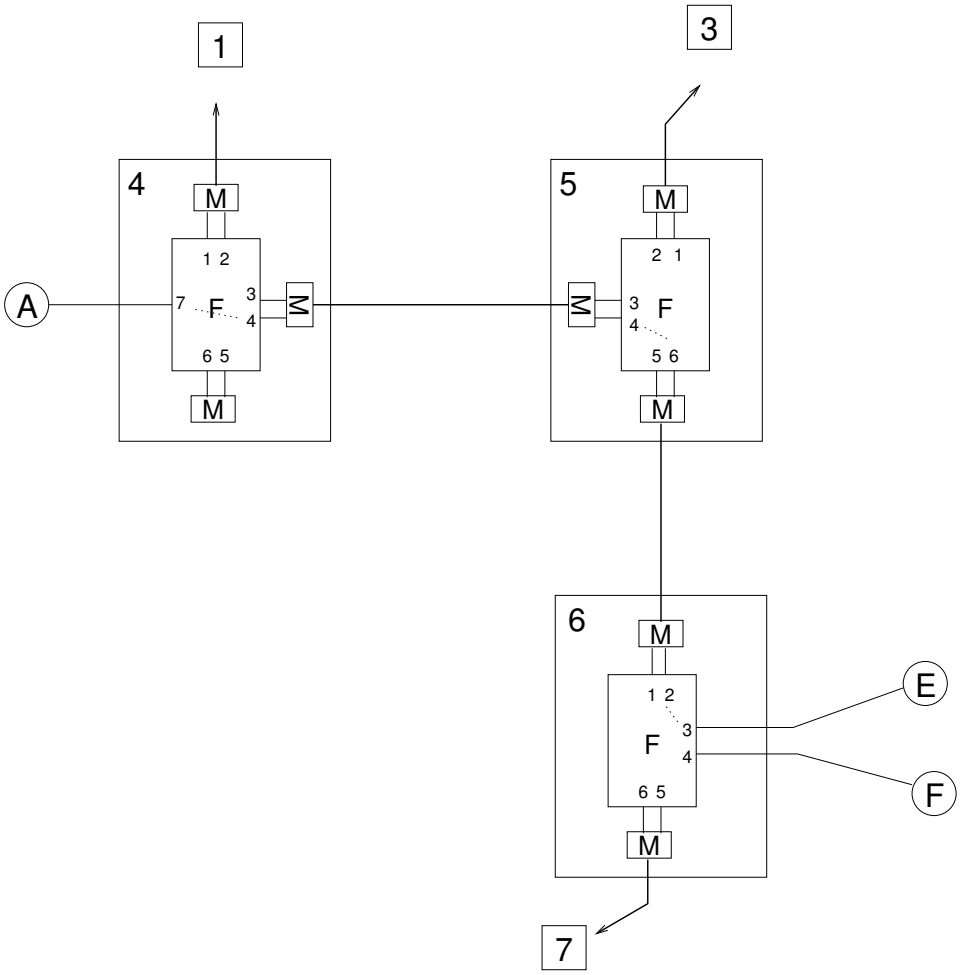
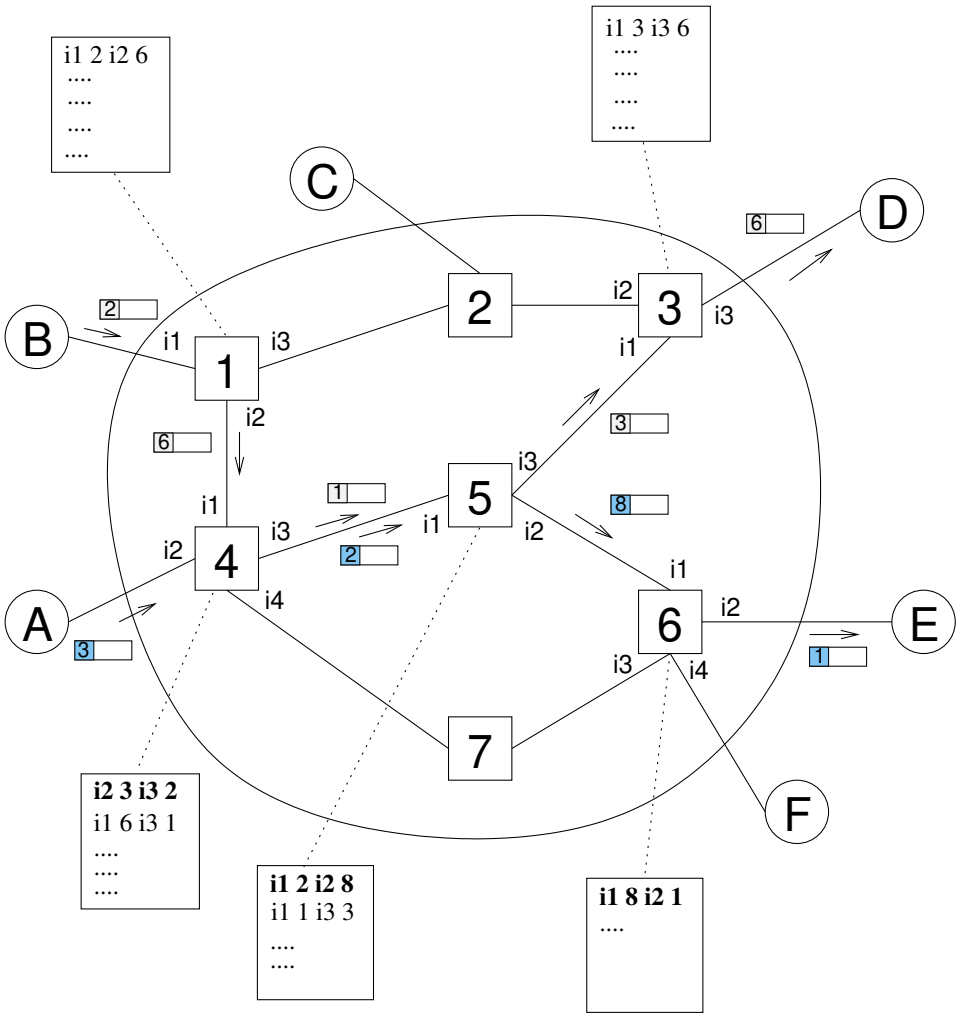


FIGURE 19 – Exemple pour la communication entre A et E



de A à E (3,2,8,1)
de B à D (2,6,1,3,6)

FIGURE 20 – Circuits virtuels

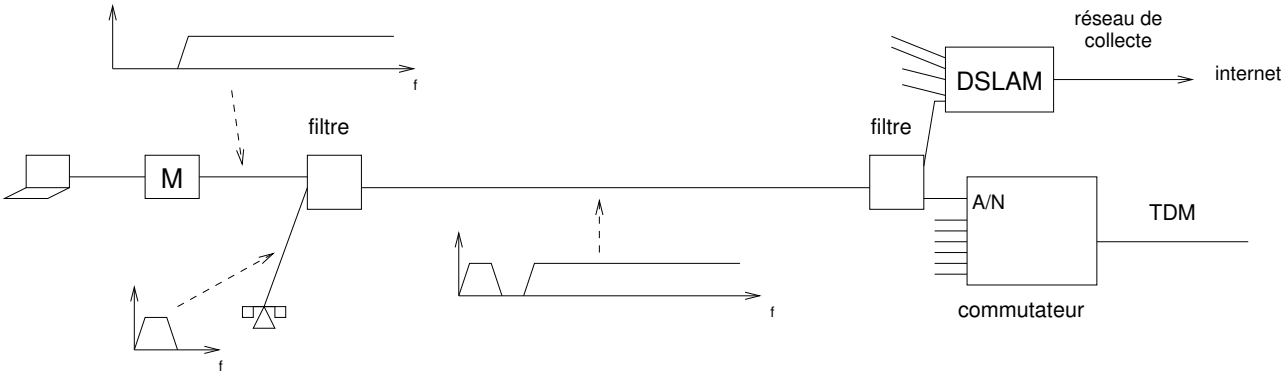


FIGURE 21 – Situation de l'ADSL

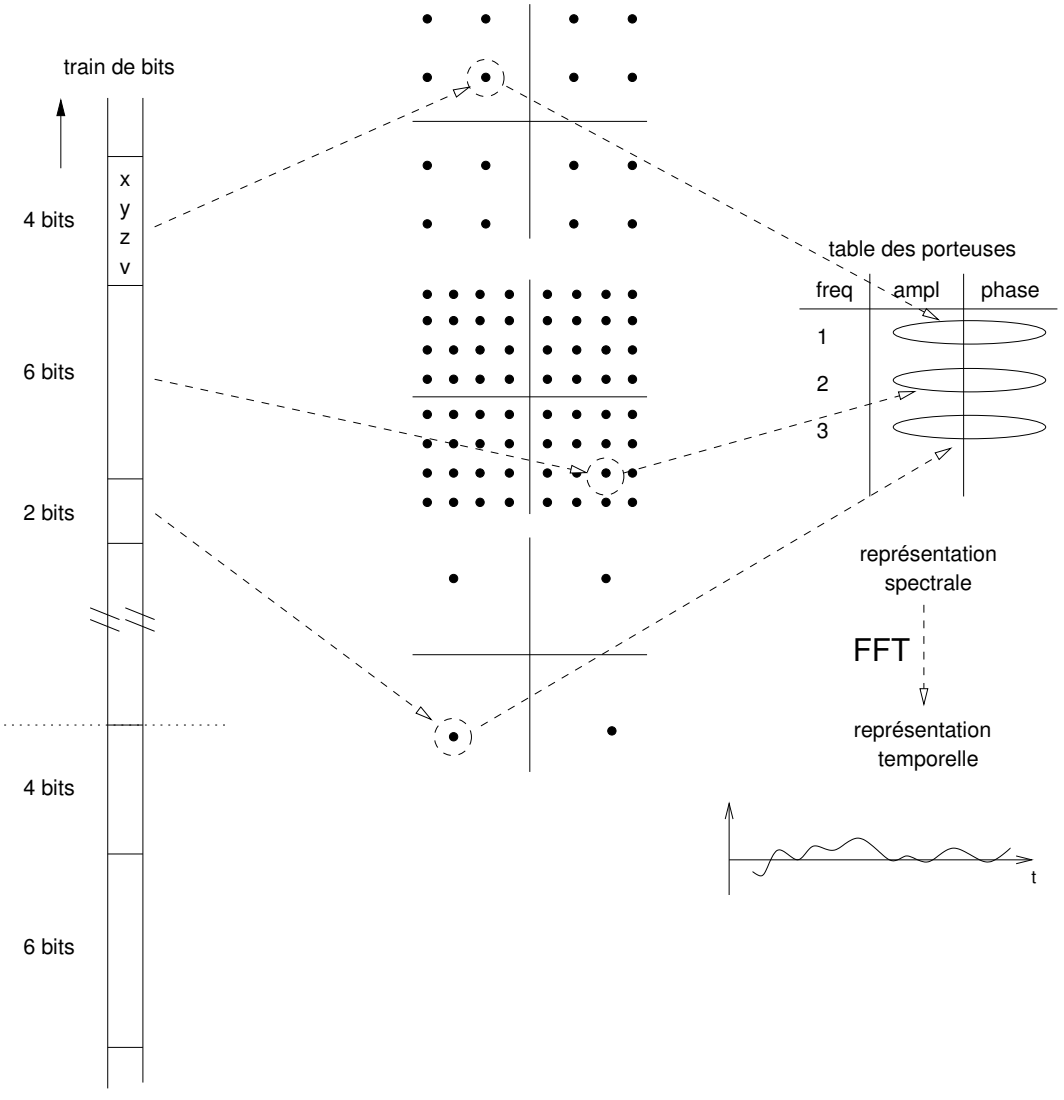


FIGURE 22 – ADSL : génération du signal