Rapport SAE S21

Réseaux Informatiques

Omar Farouk Lasfar
Muhammed Erdal
Ramzan Abdoulaev

12 juin 2025

Table des matières

Introduction

Ce rapport retrace l'ensemble des travaux réalisés dans le cadre de la SAE S21 portant sur la conception et la simulation d'un réseau informatique. Chaque séance est décrite avec les tâches effectuées, les méthodes développées et les bilans correspondants.

1 Séance 1

- Création d'un projet Git `SAE-S21`
- Création d'un document Word commun
- Création d'un groupe Discord
- Répartition des tâches

Travaux techniques

- Création de la structure `IP`

```
Tabnine | Edit | Test | Explain | Document
void afficher_ip(const ip_address_t * ip)
{
    for (unsigned int i = 0 ; i < 4 ; i++)
    {
        printf("%d", ip->paquet[i]);
        if (i < 3)
            printf(".");
    }
    printf("/%d\n", ip->masque);
}
```

- Création de la structure `Mac`

```
Tabnine | Edit | Test | Explain | Document
void afficher_mac(const mac_address_t * ma)
{
    for (unsigned int i = 0 ; i < 6 ; i++)
    {
        printf("%02X", ma->octet[i]);//
        if (i < 5)
            printf(":");
    }
    printf("\n");
}
```

– Méthode d'initialisation de la table de commutation

```c
void init_table_commutation_t(table_commutation_t *tc)
{
    tc->capacite = 8; // On garantie une capacité de 8 entrées à l'initialisation
    tc->nb_entree = 0;
    tc->entrees = calloc(tc->capacite, sizeof(entree_table_commutation_t));
    if (tc->entrees == NULL)
    {
        fprintf(stderr, "Erreur calloc() dans init_table_commutation_t !\n");
        perror("calloc : ");
        exit(EXIT_FAILURE);
    }
}
```

– Méthode de désinitialisation de la table de commutation

```c
void deinit_table_commutation_t(table_commutation_t * tc)
{
    free(tc->entrees);
    tc->entrees = NULL;
}
```

– Méthode d'ajout d'une table de commutation

```c
void ajouter_entree_table_commutation(table_commutation_t *tc, entree_table_commutation_t etc)
{
    // TODO : vérifier si l'entrée est valide par je ne sais quel moyen

    // Vérification de la taille du tableau d'entrées, l'agrandir si nécessaire
    if (tc->nb_entree == tc->capacite)
    {
        tc->capacite *= 2; // On double la capacité du tableau
        tc->entrees = realloc(tc->entrees, tc->capacite * sizeof(entree_table_commutation_t));
        // Vérification de la bonne exécution de la réallocation
        if (tc->entrees == NULL)
        {
            fprintf(stderr, "Erreur realloc() dans ajouter_entree_table_commutation !\n");
            perror("realloc : ");
            exit(EXIT_FAILURE);
        }
    }
    tc->entrees[tc->nb_entree] = etc;
    tc->nb_entree++;
}
```

– Méthode d'affichage d'une table de commutation

```c
void afficher_table_commutation(const table_commutation_t *tc)
{
    for (unsigned long i = 0 ; i < tc->nb_entree ; i++)
    {
        afficher_entree_table_commutation(&tc->entrees[i]);
    }
}
```

– Methode d'initialisation d'un Switch

```c
void init_switch_t(switch_t * sw, mac_address_t ma, unsigned short priorite_stp)
{
    sw->ma = ma;
    sw->nb_ports = 0;
    sw->priorite_stp = priorite_stp;

    // Allocation de mémoire pour une structure table_commutation_t
    sw->tc = malloc(sizeof(table_commutation_t));
    if (sw->tc == NULL)
    {
        fprintf(stderr, "Erreur dans le malloc de init_switch_t\n");
        perror("malloc : ");
        exit(EXIT_FAILURE);
    }
    init_table_commutation_t(sw->tc);
}
```

– Methode dé initialisation d'un Switch

```c
void deinit_switch_t(switch_t * sw)
{
    deinit_table_commutation_t(sw->tc);
    free(sw->tc);
    sw->tc = NULL;
}
```

– Methode d'affichage d'un Switch

```c
void afficher_switch(const switch_t * sw)
{
    printf("MAC : ");
    afficher_mac(&sw->ma);
    printf("Nombre de ports : %u\n", sw->nb_ports);
    printf("Priorité STP : %u\n", sw->priorite_stp);
    printf("Table de commutation : ");
    afficher_table_commutation(sw->tc);
}
```

Bilan de la séance 1

Nous avons rencontré des difficultés lors du choix de la structure `mac_t`. Après avoir envisagé plusieurs solutions, notamment un tableau d'entiers ou un tableau de pointeurs, nous avons finalement opté pour un tableau de `unsigned char`, plus adapté à la représentation d'une adresse MAC.

2 Séance 2

– Création de la structure `station`

```c
typedef struct
{
  ip_address_t ip;
  mac_address_t mac;
} station_t;

void afficher_station(const station_t *st);

int init_station_t(station_t *st, const char *ip, const char *mac);
```

– Méthode `afficher_station`

```c
void afficher_station(const station_t *st)
{
  printf("IP : ");
  afficher_ip_t(&st->ip);
  printf("\nMAC : ");
  afficher_mac(&st->mac);
}
```

– Méthode `init_station_t`

```c
int init_station_t(station_t *st, const char *ip, const char *mac)
{
  ip_address_t tmp_ip;
  mac_address_t tmp_ma;

  if (init_ip_address_t(&tmp_ip, ip))
  {
    st->ip = tmp_ip;
  }
  else
  {
    st->ip = (ip_address_t){{0, 0, 0, 0}, 0};
    return EXIT_FAILURE;
  }

  if (init_mac_address_t(&tmp_ma, mac))
  {
    st->mac = tmp_ma;
  }
  else
  {
    st->mac = (mac_address_t){{0, 0, 0, 0}};
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```

Bilan de la séance 2

Nous avons rencontré des difficultés à trouver des solutions aux problèmes rencontrés, ainsi qu'à répartir efficacement les tâches entre les membres du groupe. Ces blocages étaient principalement dus à une incompréhension du sujet au début du projet.

3 Séance 3

– Table de commutation changement

```c
void afficher_entree_table_commutation(const entree_table_commutation_t *etc)
{
  printf("%d:", etc->port);
  afficher_mac(&etc->ma);
}

void init_table_commutation_t(table_commutation_t *tc)
{
  tc->capacite = 8; // On garantie une capacité de 8 entrées à l'initialisation
  tc->nb_entree = 0;
  tc->entrees = malloc(tc->capacite * sizeof(entree_table_commutation_t));
  if (tc->entrees == NULL)
  {
    fprintf(stderr, "malloc dans init_table_commutation_t !\n");
    exit(EXIT_FAILURE);
  }
}
```

```c
void deinit_table_commutation_t(table_commutation_t *tc)
{
  free(tc->entrees);
  tc->entrees = NULL;
}

void ajouter_entree_table_commutation(table_commutation_t *tc,entree_table_commutation_t etc)
{
  // TODO : vérifier si l'entrée est valide par je ne sais quel moyen

  // Vérification de la taille du tableau d'entrées, l'agrandir si nécessaire
  if (tc->nb_entree == tc->capacite)
  {
    tc->capacite *= 2; // On double la capacité du tableau
    tc->entrees =
        realloc(tc->entrees, tc->capacite * sizeof(entree_table_commutation_t));
    // Vérification de la bonne exécution de la réallocation
    if (tc->entrees == NULL)
    {
      fprintf(stderr,
              "Erreur realloc() dans ajouter_entree_table_commutation !\n");
      perror("realloc : ");
      exit(EXIT_FAILURE);
    }
  }
  tc->entrees[tc->nb_entree] = etc;
  tc->nb_entree++;
}
```

```c
void afficher_table_commutation(const table_commutation_t *tc)
{
  if (tc == NULL || tc->nb_entree == 0)
  {
    printf("Table de commmutation vide.\n");
    return;
  }

  for (unsigned short i = 0; i < tc->nb_entree; i++)
  {
    afficher_entree_table_commutation(&tc->entrees[i]);
  }
}
```

– Début de la structure `réseau`

```c
// Structure pour représenter un lien
typedef struct
{
  unsigned short id1;
  unsigned short id2;
  unsigned short poids;
} lien_t;

// Enum pour le type ici
typedef enum
{
  STATION,
  SWITCH
} type_equipement_t;

// Union pour représenter un élément du réseau
typedef struct
{
  type_equipement_t type;
  unsigned short id;
  union
  {
    station_t st;
    switch_t sw;
  } contenu;
} equipement_t;
```

```c
// Structure pour représenter un réseau
typedef struct
{
  equipement_t *equipements;
  unsigned short nb_equipements;
  unsigned short index_equipement_actuel;
  unsigned short capacite_equipements;

  lien_t *liens;
  unsigned short nb_liens;
  unsigned short index_lien_actuel;
  unsigned short capacite_liens;
} reseau_t;

void init_reseau_t(reseau_t *rs);

void deinit_reseau_t(reseau_t *rs);

void afficher_reseau_t(const reseau_t *rs);

void ajouter_lien_t(reseau_t *rs, char *lien);

void ajouter_station_t(reseau_t *rs, station_t st);
/*
 * Pas sur pour celui-ci
void supprimer_lien_t(reseau_t * rs);
*/

void afficher_lien_t(const lien_t *ln);

void afficher_equipement_t(const equipement_t *eq);

void ajouter_equipement_t(reseau_t *rs, char *eq_desc);
```

Bilan de la séance 3

Une remise en question importante a eu lieu concernant la conception de la table de commutation et la structure du réseau. Les principales difficultés rencontrées portaient sur des fuites de mémoire. L'utilisation de l'outil `valgrind -leak-check=full` nous a grandement aidés à les détecter et à les corriger.

4 Séance 4

– Ajout de liens

```c
// Énumération pour les états de ports
typedef enum
{
  ROOT,
  DESIGNATED,
  BLOCKED
} etat_port_t;

// Structure pour un switch
typedef struct
{
  mac_address_t ma;
  unsigned short nb_ports;
  etat_port_t * etat_ports;
  // TODO ajouter état + rôle des ports aussi + un truc pour stocker/gérer les
  // messages
  unsigned short priorite_stp;
  table_commutation_t tc;
} switch_t;
```

– Ajout du fichier config.c

```c
void charger_configuration(reseau_t *rs, const char *path)
{
  // Charger les données contenues dans le fichier de configuration se situant
  // dans path dans le réseau passé en argument
  FILE *fichier = fopen(path, "r");

  if (fichier == NULL)
  {
    perror("fopen (charger_configuration)");
    deinit_reseau_t(rs);
    exit(EXIT_FAILURE);
  }

  char ligne_buffer[TAILLE_BUFFER];

  if (fgets(ligne_buffer, sizeof(ligne_buffer), fichier) != NULL)
  {
    // Parsing du nombre d'équipements et du nombre de liens
    if (sscanf(ligne_buffer, "%hu %hu", &rs->nb_equipements, &rs->nb_liens) !=
        2)
    {
      fprintf(stderr, "Erreur : fichier de configuration malformé\n");
      deinit_reseau_t(rs);
      fclose(fichier);
      exit(EXIT_FAILURE);
    }
```

```c
    // Ajout des équipements
    for (size_t i = 0; i < rs->nb_equipements; i++)
    {
        if (fgets(ligne_buffer, sizeof(ligne_buffer), fichier) != NULL)
        {
            ajouter_equipement_t(rs, ligne_buffer);
        }
        else
        {
            fprintf(stderr, "Erreur : nombre de liens incorrect dans le fichier\n");
            deinit_reseau_t(rs);
            fclose(fichier);
            exit(EXIT_FAILURE);
        }
    }

    // Ajout des liens
    for (size_t i = 0; i < rs->nb_liens; i++)
    {
        if (fgets(ligne_buffer, sizeof(ligne_buffer), fichier) != NULL)
        {
            ajouter_lien_t(rs, ligne_buffer);
        }
        else
        {
            fprintf(stderr, "Erreur : nombre de liens incorrect dans le fichier\n");
            deinit_reseau_t(rs);
            fclose(fichier);
            exit(EXIT_FAILURE);
        }
    }
}

fclose(fichier);
```

```c
void afficher_trame(const trame_ethernet_t *trame)
{
    printf("Adresse mac destination : ");
    afficher_mac(&trame->dest);
    printf("/n"); // adresse mac dest
    printf("Adresse mac source : ");
    afficher_mac(&trame->src);
    printf("/n"); //adresse mac src
    printf("type : 0x%04x\n",trame->type);// adresse type
    printf("\nTaille données: %u octets\n", trame->taille_donnees);
    for (int i = 0; i < trame->taille_donnees; i++)
    {
        printf("%02X ", trame->donnees[i]);
        if ((i + 1) % 16 == 0)
            printf("\n");
    }
    printf("\n");
}
```

Bilan de la séance 4

Cette séance a été très positive. Nous avons réussi à implémenter l'analyse (parsing) du fichier de configuration ainsi que l'initialisation automatique du réseau à partir de ce fichier. M. Wemmert nous a apporté son aide sur plusieurs points, notamment concernant la structure des trames, ce qui nous a permis de mieux orienter notre travail.

Travail en dehors des séances

Nous avons :

- Modifié la structure de la trame et son affichage
- Implémenté le transit d'une trame dans le réseau (gestion des erreurs : MAC inexistante, station invalide, etc.)
- Rempli dynamiquement les tables de commutation avec ces trames, simulant ainsi le comportement réel d'un switch

Conclusion

Cette SAE fut complexe, avec du retard et certaines parties à refaire. Malgré cela, l'organisation et la répartition des tâches ont été respectées, et nous avons réussi à atteindre le début de la dernière phase : le protocole STP.