CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

Grenoble

Rapport de Stage

Métrologie Temps Réel sur un Réseau Gigabit Ethernet

Annexes

Stage du 17 juin au 31 aout 2002

Maîtres de stage
M. Daniel Gueniche
M. Laurent Neiger

Stagiaires
Olivier CHAUSSINAND
Elise GALLENSTEIN
Gaston MASSALA

Table des matières

Collecte

I. Analyseur PERL

```
# ------------------------------------------------------------------------------
#                   ANALYSEUR.PL
#
# Programme d'analyse des trames icmp, tcp et udp donnees en entree standard.
# Il permet de retirer les informations :
#   - Num de protocole
#   - Date
#   - Adresse Source
#   - Port Source
#   - Adresse Destinataire
#   - Port Destinataire
#   - Flag (pour les trames tcp)
#   - Taille des donnees (en octets)
#
# Il est adapte pour utiliser les trames de tcpdump:
#   - tcpdump -nnv udp or tcp or icmp
#
#-------------------------------------------------------------------------------

#!/usr/bin/perl
# use FileHandle;

# ------------------------------------------------------------------------------
# Variable permettant de choisir le mode d'execution
#   0 : Programme seul
#   1 : Avec lancement de tcpdump integre
#   2 : Avec lancement du collecteur integre
#   3 : Avec lancement de tcpdump et du collecteur integre
# ------------------------------------------------------------------------------
$mode = 0;

if ($mode == 1 || $mode == 3)
{
  ($pid = open(PIPE, "tcpdump -nnv udp or tcp or icmp |")) or die "Error: $!\n";
  (kill 0, $pid) or die "ps invocation failed";
}

while (1)
{

# ---------------------------------------------------------------------
# Initialisation des variables devant recevoir les infos des paquets
# ---------------------------------------------------------------------
  $paquet="";
  $proto="";
  $date="";
  $source="";
  $dest="";
  $adrSour="";
  $adrDest="";
  $portDest=0;
  $portSour=0;
  $flag="\.";
  $nboct="0";

# select((select(STDIN), $| = 1)[0]);
# autoflush STDIN 1;

# ------------------------------------------------
# Lecture des paquets sur l'entree standard
# Et supression du retour chariot
# ------------------------------------------------
#  $paquet = <STDIN>;

if ($mode == 0 || $mode == 2)
{
```

```
  chomp($paquet = <STDIN>);
}
else
{
  chomp($paquet = <PIPE>);
}

#  printf("$paquet \n");

# ------------------------------------------------------------------------------
# Test pour verifier qu'il y a quelque chose a l'entree
# ceci est necessaire lorsqu'on lit dans un fichier pour lui eviter de boucler
# ------------------------------------------------------------------------------
  if ($paquet ne "")
  {
    ($poubelle, $poubelle, $poubelle, $tmp) = split (/:/, $paquet,4);
#    printf ("Tmp apres : : $tmp\n");

    # --------------------------------------------------------------------------
    # On test si on sait traiter les paquets recu:
    #   - commancant par une date
    #   - ayant des informations derriere les deux points
    # S'il ne correspond pas a des paquets traitables il est inserer dans un
    # fichier analyseur.log
    # --------------------------------------------------------------------------
    if (!(substr($paquet,0,15)=~/\d\d:\d\d:\d\d.\d\d\d\d\d\d\d/)||$tmp eq ""|| !($tmp=~/ len /))
    {
      open(FILE, ">>analyseur.log");
      if (substr($paquet, 0, 15) =~ /\d\d:\d\d:\d\d.\d\d\d\d\d\d/)
      {
        print FILE "MAUVAIS FORMAT:\n $paquet\n";
      }
      else
      {
        print FILE "$paquet\n";
      }
      close FILE;
    }
    else
    {

      # ----------------------------------------------------------
      # Recuperation des donnees communes aux paquets :
      #   - Date : hh:mm:ss
      #   - Source : XXX.XXX.XXX.XXX.YYY
      #   - Dest :   XXX.XXX.XXX.XXX.YYY
      # avec XXX.XXX.XXX.XXX adresse Ip et YYY port
      # sauf pour les paquets icmp qui n'ont pas de ports
      #   - NbOct : se trouvant en fin de chaine apres le champ len
      # ----------------------------------------------------------
      ($date, $tmp) = split (/\./,$paquet,2);
      # printf("Date : $date\n");
      ($poubelle, $source, $poubelle, $tmp) = split (/ /,$tmp, 4);
      ($dest, $tmp) = split(/:/,$tmp,2);
      ($poubelle, $nboct) = split(/ len /,$tmp);
      ($nboct, $poubelle) = split(/\)/, $nboct);
      if ($nboct eq "")
      {
        $nboct = 0;
      }
      #printf("Source : $source\nDest : $dest\nNbOct : $nboct\nReste : $tmp\n");

      # ----------------------------------------------------------
      # Test pour savoir si le paquet recupere est de format icmp
      # ----------------------------------------------------------
      if (substr($tmp, 1, 4) eq "icmp")
      {
        $proto = 1;
        $adrSour = $source;
        $adrDest = $dest;

        # ----------------------------------------------------------
        # Recuperation de la fin des infos du paquet et analyse
        # pour connaitre le type de paquet icmp auquel il correspond
        # Le type du paquet est mis a la place du port destinataire
        # ----------------------------------------------------------
        ($poubelle, $poubelle, $tmp) = split (/ /,$tmp, 3);
```

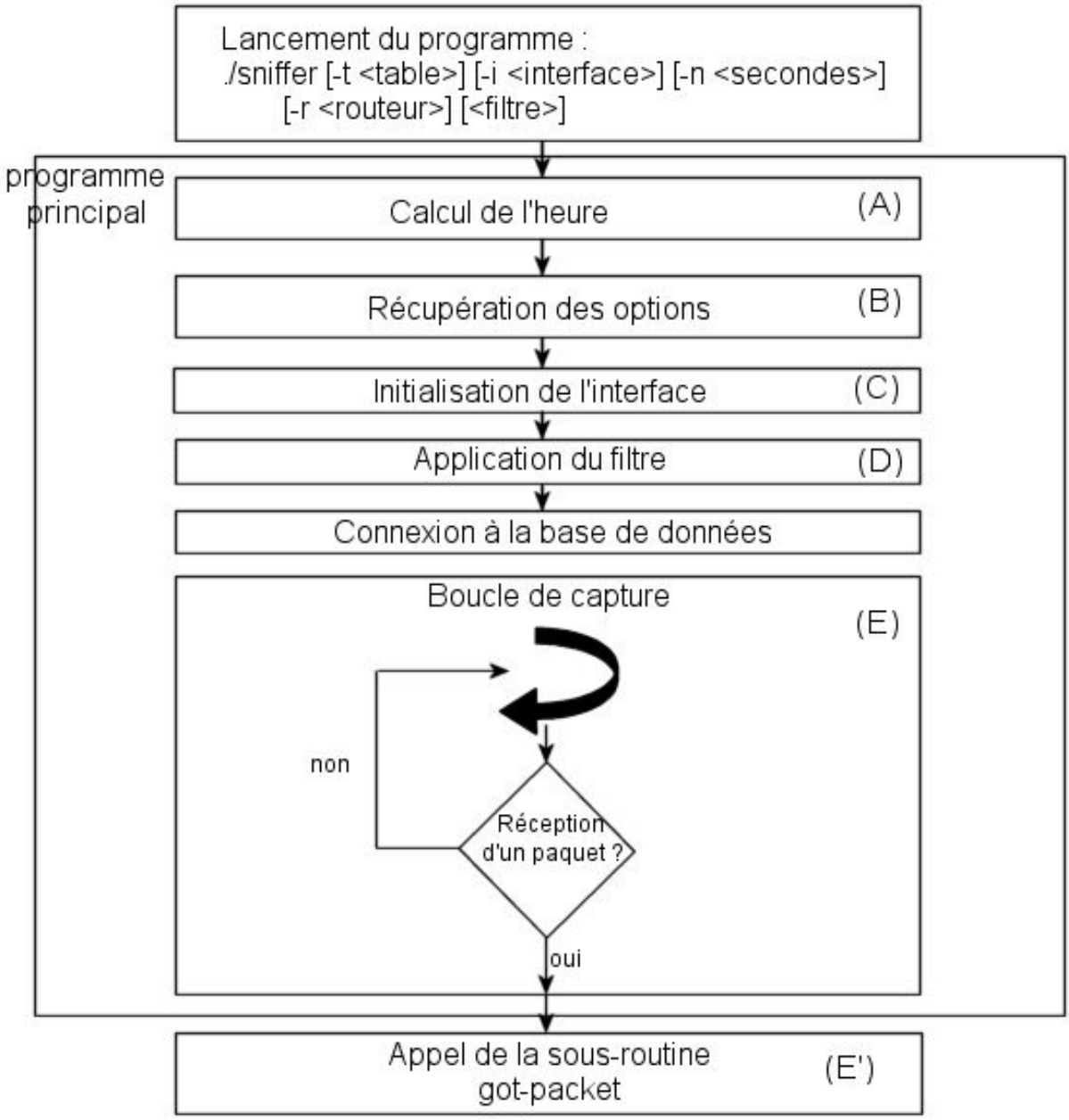
```
  #      printf("Tmp : $tmp");
       if (substr($tmp, 0, 10) eq "echo reply")
       {
         $portDest = 0;
       }
       elsif ($tmp =~ /unreachable/)
       {
         $portDest = 3;
       }
       elsif ($tmp =~ /source quench/)
       {
         $portDest = 4;
       }
       elsif (substr($tmp, 0, 8) eq "redirect")
       {
         $portDest = 5;
       }
       elsif (substr($tmp, 0, 12) eq "echo request")
       {
         $portDest = 8;
       }
       elsif ($tmp =~ /time exceeded/)
       {
         $portDest = 11;
       }
       else
       {

         # -------------------------------------------------------
         # Par default le type est mis a 256 (num non utilise)
         # -------------------------------------------------------
         $portDest = 256;
         open(FILE, ">>analyseur.log");
         print FILE "NOUVEAU TYPE D ICMP:\n $paquet\n";
         close FILE;
       }
#       printf ("Paquet ICMP!!!!\n");
     }

     else
     {

       # -------------------------------------------------------
       # Dans ce cas les paquets sont des udp ou tcp
       # -------------------------------------------------------

       # ------------------------------------------------------------------
       # Recuperation des adresse et port  a partir des info source et dest
       #   - adrSour : Adresse source (XXX.XXX.XXX.XXX)
       #   - adrDest : Adresse destination (XXX.XXX.XXX.XXX)
       #   - portSour : Port source (XXX)
       #   - portDest : Port source (XXX)
       # ------------------------------------------------------------------
       ($adr1, $adr2, $adr3, $adr4,$portSour) = split (/\./,$source);
       $adrSour = $adr1."\.".$adr2."\.".$adr3."\.".$adr4;
       ($adr1, $adr2, $adr3, $adr4,$portDest) = split (/\./,$dest);
       $adrDest = $adr1."\.".$adr2."\.".$adr3."\.".$adr4;
#       printf("Adresse Source : $adrSour\nPort Source : $portSour\n");
#       printf("Adresse Dest : $adrDest\nPort Dest : $portDest\n");

       # ------------------------------------------------------------------
       # Les paquets tcp sont caracterises par la presence de la fenetre
       # indiquee par l'info win dans les paquets
       # On recherche si le champ win est present dans le paquet
       # ------------------------------------------------------------------
       ($poubelle, $flagtmp, $tmp2) = split (/ /,$tmp,3);
#       printf("Tmp2 : $tmp2 \n");
       if ($tmp2 =~ / win /)
       {

         # ------------------------------------------------------------------
         # Ces paquets sont du type tcp (6) on recupere donc le flag ainsi que
         # la taille des donnees qui se trouve entre parentheses
         # ------------------------------------------------------------------
         $proto = 6;
         ($flag) = split (/ /,$flagtmp);
#         printf ("Paquet TCP!!!!\n");
```

```
        }
        else
        {

          # ------------------------------------------------------------------
          # Ces paquets sont du type udp (17)
          # ------------------------------------------------------------------
          $proto = 17;

#          printf ("Paquet UDP!!!!\n");
        }
      }
#      printf("Proto : $proto\nDate : $date\nAdresse Source : $adrSour\nPort");
#      printf(" Source : $portSour\nAdresse Dest : $adrDest\nPort Dest : ");
#      printf("$portDest\nFlag : $flag\nNbOct : $nboct\n");
#      $| = 1;
#     select((select(STDOUT), $| = 1)[0]); # autoflush STDOUT 1;

    ($adrS1, $adrS2, $adrS3, $adrS4) = split (/\./,$adrSour);
    ($adrD1, $adrD2, $adrD3, $adrD4) = split (/\./,$adrDest);
    if ($proto eq "" || $date eq "" || $adrS1 eq "" || $adrS2 eq "" || $adrS3 eq "" || $adrS4
eq "" || $adrD1 eq "" || $adrD2 eq "" || $adrD3 eq "" || $adrD4 eq "" || $portSour eq "" ||
$portDest eq "" || $flag eq ""  || $nboct eq "")
    {
      open(FILE, ">>analyseur.log");
      print FILE "CHAMP VIDE:\n $paquet\n";
      print FILE "$proto $date $adrSour $portSour $adrDest $portDest $flag $nboct\n";
      close FILE;
    }
    else
    {
    # ------------------------------------------------------------------
    # Sortie ecran des informations necessaires :
    # Protocole Date Adresse_Source Port_Source Adresse_Destinataire
    # Port_Destinataire Flag Nombre_Octet
    # ------------------------------------------------------------------
      print("$proto $date $adrSour $portSour $adrDest $portDest $flag $nboct\n");
    }
    }
  }
}

close(PIPE);
```

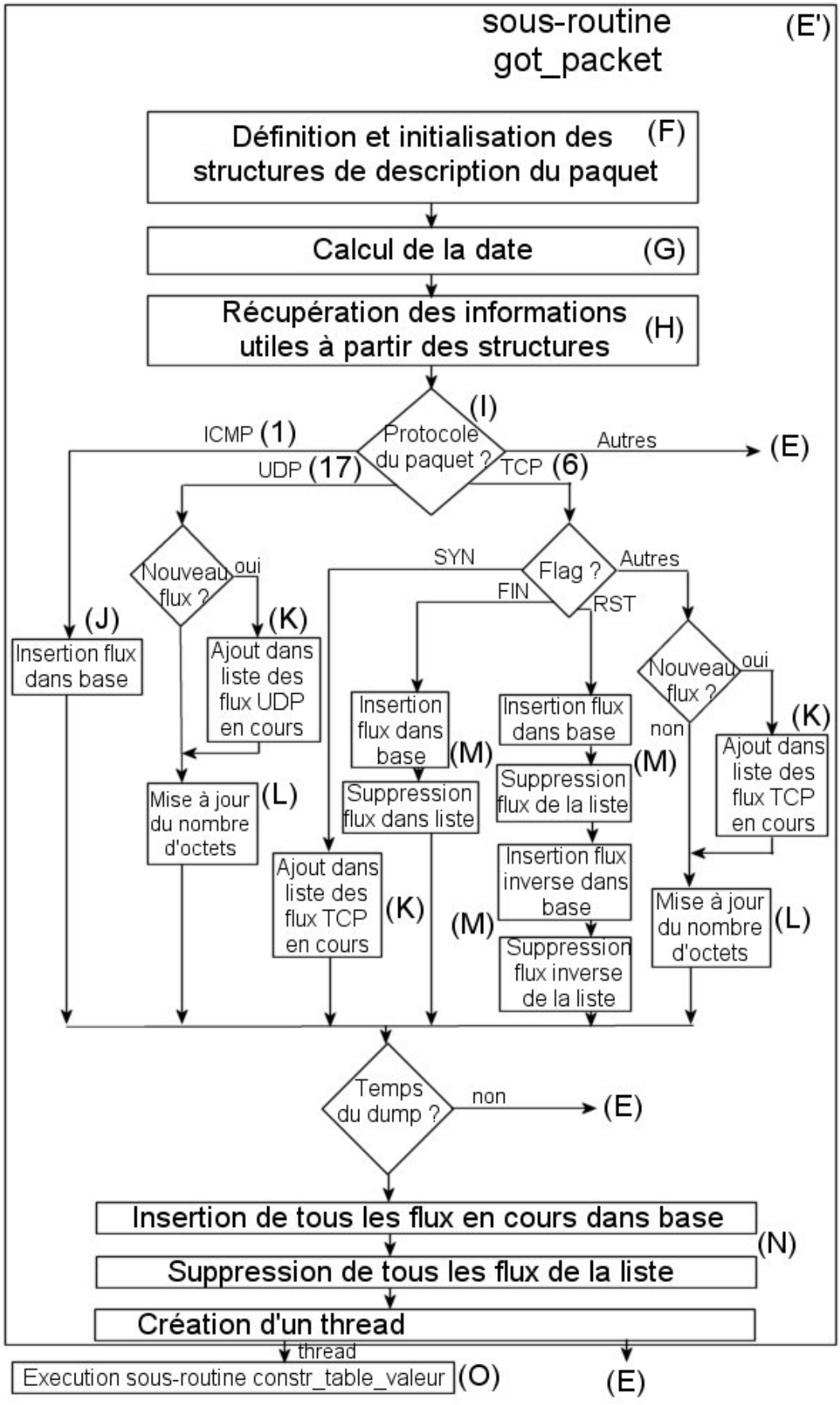
II. Programme de collecte

A. Architecture du programme

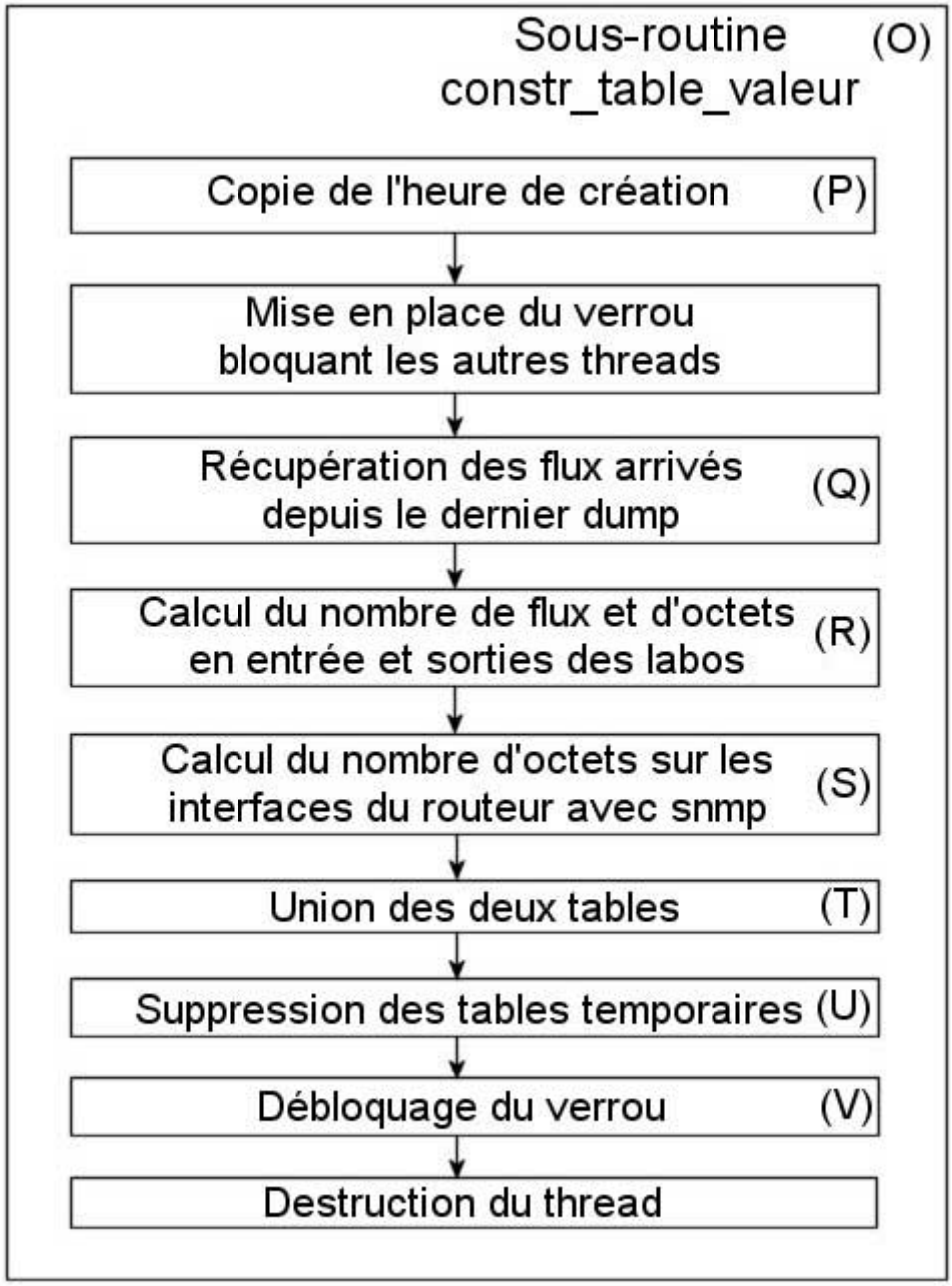
1. Lancement, initialisation et boucle de capture du programme



2. Traitement des paquets



3. Réalisation des calculs dans la base de données



B. Code du programme

```
/*****************************************************************************
* sniffer.c                                                                  *
* Programme issu du projet GEO                                               *
* utilise la libpcap et la lib mysql pour remplir une base de donnees avec les     *
* flux circulant sur le reseau.                                              *
*                                                                            *
*****************************************************************************/


#define _BSD_SOURCE 1
#define MAXSNMP 4294967295

#define LOG "/root/elise/pcap/log-sniffer"

#include <pthread.h>
#include <pcap.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <net/if.h>
#include <netinet/if_ether.h>
#include <netinet/tcp.h>
#include <sys/types.h>
#include <sys/time.h>
#include <time.h>
#include <mysql.h>
#include "interface.h"

/*****************************************************************************
* on a besoin ici de declarer 3 structures contenant les infos des entetes         *
* ethernet, ip et tcp pour pouvoir exploiter les donnees retournees par la         *
* libpcap sur chaque paquet capture.                                         *
*****************************************************************************/


/* Ethernet header */
struct sniff_ethernet
{
        u_char  ether_dhost[ETHER_ADDR_LEN];    /* Destination host address */
        u_char  ether_shost[ETHER_ADDR_LEN];    /* Source host address */
        u_short ether_type;                     /* IP? ARP? RARP? etc */
};

/* IP header */
struct sniff_ip
{
        #if BYTE_ORDER == LITTLE_ENDIAN
                u_int   ip_hl:4,        /* header length */
                        ip_v:4;         /* version */
        #if BYTE_ORDER == BIG_ENDIAN
                u_int   ip_v:4,         /* version */
                        ip_hl:4;        /* header length */
        #endif
        #endif /* not _IP_VHL */
        u_char  ip_tos;                 /* type of service */
        u_short ip_len;                 /* total length */
        u_short ip_id;                  /* identification */
        u_short ip_off;                 /* fragment offset field */
        #define IP_RF 0x8000            /* reserved fragment flag */
        #define IP_DF 0x4000            /* dont fragment flag */
        #define IP_MF 0x2000            /* more fragments flag */
        #define IP_OFFMASK 0x1fff       /* mask for fragmenting bits */
        u_char  ip_ttl;                 /* time to live */
        u_char  ip_p;                   /* protocol */
        u_short ip_sum;                 /* checksum */
        struct  in_addr ip_src,ip_dst;  /* source and dest address */
};

/* TCP header */
struct sniff_tcp
{
```

```
        u_short th_sport;                       /* source port */
        u_short th_dport;                       /* destination port */
        tcp_seq th_seq;                         /* sequence number */
        tcp_seq th_ack;                         /* acknowledgement number */
        #if BYTE_ORDER == LITTLE_ENDIAN
                u_int   th_x2:4,                /* (unused) */
                        th_off:4;               /* data offset */
        #endif
        #if BYTE_ORDER == BIG_ENDIAN
                u_int   th_off:4,               /* data offset */
                        th_x2:4;                /* (unused) */
        #endif
        u_char  th_flags;
        #define TH_FIN  0x01
        #define TH_SYN  0x02
        #define TH_RST  0x04
        #define TH_PUSH 0x08
        #define TH_ACK  0x10
        #define TH_URG  0x20
        #define TH_ECE  0x40
        #define TH_CWR  0x80
        #define TH_FLAGS        (TH_FIN|TH_SYN|TH_RST|TH_ACK|TH_URG|TH_ECE|TH_CWR)
        u_short th_win;                         /* window */
        u_short th_sum;                         /* checksum */
        u_short th_urp;                         /* urgent pointer */
};

/*****************************************************************************
* on declare un structure de type flux qui va stocker les differentes infos  *
* necessaires concernant un flux TCP ou UDP                                  *
*****************************************************************************/

typedef struct
{
      int proto;
        char heureDebut[9];
        char heureFin[9];
        char adrSrc[16];
        int portSrc;
        char adrDest[16];
        int portDest;
        int nbOct;
} flux;


/*****************************************************************************
* on definit une structure qui va permettre de reconstruire les differents flux  *
* TCP ou UDP qui circulent sur le reseau. on a donc une structure qui permet de  *
* gerer l'ensemble des flux TCP ou UDP en cours. un flux est retire de la liste  *
* pour 2 raisons : soit parce qu'il est termine (reception d'un paquet portant   *
* les flags F ou R dans le cas d'un flux TCP, soit parce qu'on dumpe les flux en *
* cours pour privilegier le traitement en temps reel.                            *
*****************************************************************************/

typedef struct _l_flux
{
        flux element;
        struct _l_flux *suivant;
} l_flux;


/*****************************************************************************
* on definit une structure pour stocker les precedentes valeurs de           *
* l'interrogation SNMP du routeur sur ifInOctets et ifOutOctets pour chaque  *
* port.                                                                      *
*****************************************************************************/

typedef struct
{
      unsigned long in;
      unsigned long out;
} s_old;


/*****************************************************************************
* declaration des variables globales                                         *
*****************************************************************************/
```

```
l_flux *teteTCP;        // tete de la liste TCP
l_flux *teteUDP;        // tete de la liste UDP
time_t temps;           // variables pour le calcul du temps
struct tm *infos;
char tdate[20];
int min, hour, sec, hour_new, min_new, sec_new, hour_old, min_old, sec_old, secd;
MYSQL *db ;             // connexion a la base de donnees
MYSQL *db_thread ;      // connexion a la base de donnees pour
                        // les threads
char table[20] = "flots";           // nom de la table dans laquelle
                                    //on insere les flux
char date_new[30], date_old[30];    // date de dump
FILE *log;              // fichier de log

s_old old[50];          // stocke les valeurs precedentes de snmp
int dump1;              // pour savoir si on vient de lancer le programme

static pthread_mutex_t mutex; // permet de ne pas avoir 2 threads qui ecrivent
                              // dans la base en meme temps
static int nb_attente = 0;    // juste pour verifier s'il y a des threads qui
                              // se marchent dessus

char adr_routeur[16] = "xxx.xxx.1.103";         // adresse ip du routeur a
                                                // interroger pour le snmp

/*****************************************************************************
* Affiche la liste des connexions TCP en cours (utilise pour le debug)         *
*****************************************************************************/

void afficherListe()
{
        l_flux *courant;

        courant = teteTCP;
        while (courant != NULL)
      {
            printf("heureDebut=%s heureFin=%s adrSrc=%s portSrc=%d adrDest=%s portDest=%d
nbOct=%d\n", courant->element.heureDebut, courant->element.heureFin, courant->element.adrSrc,
courant->element.portSrc, courant->element.adrDest, courant->element.portDest, courant-
>element.nbOct);
                courant = courant->suivant;
        }
        free(courant);
}


/*(K)*/
/*****************************************************************************
* Insere un nouveau flot UDP ou TCP dans la liste, insertion en tete          *
*****************************************************************************/

void inserer1flux(flux elt)
{
        l_flux *nouveau;
        nouveau = (l_flux*)malloc(sizeof(l_flux));
        nouveau->element = elt;
      if (elt.proto == 6)
      {
            nouveau->suivant = teteTCP;
            teteTCP = nouveau;
      } else
      {
            nouveau->suivant = teteUDP;
                teteUDP = nouveau;
      }
        nouveau = NULL;
        free(nouveau);
}

/*(L)*/
/*****************************************************************************
* ajoute des octets a une connexion TCP ou UDP en cours lors de la reception *
* d'un paquet correspondant a cette connexion                                 *
*****************************************************************************/
```

```
void mettre_a_jour_nbOct(flux elt)
{
        l_flux *courant;
      if (elt.proto == 6)
            courant = teteTCP;
      else
            courant = teteUDP;
        while (!(  courant == NULL
               || (!strcmp(courant->element.adrSrc, elt.adrSrc)
               && courant->element.portSrc == elt.portSrc
               && !strcmp(courant->element.adrDest, elt.adrDest)
               && courant->element.portDest == elt.portDest)))
      {
                courant = courant->suivant;
        }
        if (courant != NULL)
      {
                courant->element.nbOct += elt.nbOct;
        } else
      {
                inserer1flux(elt);
        }
        courant = NULL;
        free(courant);
}

/*****************************************************************************
* on se connecte au serveur mysql et on cree la table des flux si besoin      *
*****************************************************************************/

int connecter_base()
{
        char inter[1000];     // chaine contenant la requete a envoyer a la base

        db = mysql_init(NULL);
        if (mysql_real_connect(db, "localhost", "netflow", "netflow", "netflow", 0, NULL, 0)
== NULL)
      {
                printf("mysql_real_connect() failed\n");
            log = fopen(LOG, "a");
                fprintf(log, "%s : mysql_real_connect() failed\n", date_old);
            fclose(log);
                return -1;
        }

      /* on cree la table de stockage des flux si elle n'existe pas */
        sprintf(inter, "create table if not exists %s (numflot bigint unsigned not null
auto_increment primary key, ipsrc1 tinyint unsigned, ipsrc2 tinyint unsigned, ipsrc3 tinyint
unsigned, ipsrc4 tinyint unsigned, ipdst1 tinyint unsigned, ipdst2 tinyint unsigned, ipdst3
tinyint unsigned, ipdst4 tinyint unsigned, portdst mediumint unsigned, starttime datetime,
endtime datetime, taille bigint unsigned, typprot tinyint unsigned)", table);
        if (mysql_query(db, inter) != 0)
      {
            log = fopen(LOG, "a");
                fprintf(log, "%s : process_query() create table failed\n", date_new);
            fclose(log);
                printf("process_query() create table failed\n");
                return -1;
        }
        return 0;
}

/*****************************************************************************
* on se connecte au serveur mysql pour un thread                              *
*****************************************************************************/

int connecter_base_thread()
{
        char inter[1000];
        db_thread = mysql_init(NULL);
        // MYSQL *mysql_real_connect(MYSQL *mysql, const char *host, const char *user, const
char *passwd, const char *db, unsigned int port, const char *unix_socket, unsigned int
client_flag)

        if (mysql_real_connect(db_thread, "localhost", "netflow", "netflow", "netflow", 0,
NULL, 0) == NULL)
      {
```

```
                printf("mysql_real_connect() failed\n");
            log = fopen(LOG, "a");
                fprintf(log, "%s : mysql_real_connect() failed\n", date_old);
            fclose(log);
                return -1;
        }
        return 0;
}

/*****************************************************************************
* on quitte la base de donnees                                               *
*****************************************************************************/

void fermer_base() {
        mysql_close(db);
}

/*****************************************************************************
* on quitte la base de donnees pour un thread                                *
*****************************************************************************/

void fermer_base_thread() {
        mysql_close(db_thread);
}

/*****************************************************************************
* remplit la table necessaire pour l'analyse par labo realise par le         *
* programme phip. on a besoin de passer par des tables intermediaires pour le  *
faire. cette fonction est executee apres chaque dump par un thread.          *
*****************************************************************************/

void * constr_table_valeur(void * arg)
{
        char inter[1000];     // requete SQL
      int connect;            // resultat de la connexion
      MYSQL_FIELD *field;
      MYSQL_ROW row;

      int num;          // numero du tube ouvert pour la commande snmptable
      char car;         // caractere lu dans le tube
      char chaine[500]; // chaine stockant la ligne en cours de
                        // traitement de la reponse snmp
      int i;
      unsigned long in, out, deltain, deltaout;      // variables pour le
                                          // calcul de snmp
      int num_port;           // numero du port du routeur

      char date_perso_old[30];      // on conserve les dates du moment de
                              // lancement du thread
      char date_perso_new[30];      // au cas ou il prenne du retard
      int hour_perso, min_perso, sec_perso;

      int jour_semaine;       // pour savoir quel jour on supprime la table
      int jour_semaine_old;   // flot


/*(P)*/
      strcpy(date_perso_old, date_old);
      strcpy(date_perso_new, date_new);

      /*-- on bloque tous les threads qui pourraient arriver pendant le
      traitement pour eviter qu'ils se marchent dessus */

//    printf("EN ATTENTE : %d\n", nb_attente);
      nb_attente ++;
      pthread_mutex_lock (&mutex);

//    printf("JE SUIS AWARE !!!!\n");
        connect = connecter_base_thread();

      /*-- on deplace la fenetre des flots conserves cad qu'on garde ceux
      des 24 dernieres heures (86400 sec)*/

      sprintf(inter, "DELETE FROM valeurs_labo where timestamp <
FROM_UNIXTIME(UNIX_TIMESTAMP(\"%s\") - 86400)", date_perso_new);
      if (mysql_query(db_thread, inter) != 0)
      {
```

```
            log = fopen(LOG, "a");
                fprintf(log, "%s : process_query() delete from valeurs_labo trop vieux
failed\n", date_new);
            fclose(log);
                printf("process_query() delete from valeurs_labo trop vieux failed\n");
        }
        sprintf(inter,"DELETE FROM %s where endtime < FROM_UNIXTIME(UNIX_TIMESTAMP(\"%s\") -
86400)", table, date_perso_new);
      if (mysql_query(db_thread,inter) != 0)
      {
            log = fopen(LOG, "a");
                fprintf(log, "%s : process_query() delete from flots trop vieux failed\n",
date_new);
            fclose(log);
                printf("process_query() delete from flots trop vieux failed\n");
        }

/*(Q)*/
      /*-- on recupere tous les flux recus depuis le dernier dump et on les met
      dans une table temporaire*/

        sprintf(inter, "INSERT INTO flotsrecents SELECT numflot, ipsrc1, ipsrc2, ipsrc3,
ipsrc4, ipdst1, ipdst2, ipdst3, ipdst4, taille FROM %s WHERE endtime BETWEEN
FROM_UNIXTIME(UNIX_TIMESTAMP(\"%s\")+1) AND \"%s\"", table, date_perso_old, date_perso_new) ;
        if (mysql_query(db_thread, inter) != 0)
      {
            log = fopen(LOG, "a");
                fprintf(log, "%s : process_query() insertion into flotsrecents failed\n",
date_new);
            fclose(log);
                printf("process_query() insertion into flotsrecents failed\n");
        }

/*(R)*/
      /*-- on associe a chaque ip source un labo ce qui permet de compter le
      nombre d'octets et le nombre de flux correspondants au labo. au cas
      ou l'ip ne correspond pas a un labo, on utilise le labo CICG */

        sprintf(inter, "INSERT INTO tablein SELECT IFNULL(nomlabo, \"CICG\") , count(*)
flotin, sum(taille) octetin FROM flotsrecents F LEFT JOIN ipreseau R ON R.ipsrc1 = F.ipdest1
and R.ipsrc2 = F.ipdest2 and R.ipsrc3 = F.ipdest3 GROUP BY R.nomlabo");
        if (mysql_query(db_thread, inter) != 0)
      {
            log = fopen(LOG, "a");
                fprintf(log, "%s : process_query() insertion into tablein failed\n",
date_new);
                fclose(log);
                printf("process_query() insertion into tablein failed\n");
        }

      /*-- on recroise le resultat precedent avec la table des adresses des
      labos pour avoir une entree pour tous les labos y compris ceux qui
      n'ont pas eu de flots en entree */

        sprintf(inter, "INSERT INTO tableinbis SELECT distinct(I.nomlabo),flotin, octetin FROM
ipreseau I LEFT JOIN tablein T ON I.nomlabo=T.nomlabo");

        if (mysql_query(db_thread, inter) != 0)
      {
            log = fopen(LOG, "a");
            fprintf(log, "%s : process_query() insertion into tableinbis failed\n", date_new);
                fclose(log);
                printf("process_query() insertion into tableinbis failed\n");
        }

      /*-- on fait la meme chose pour les flots en sortie */
        sprintf(inter, "INSERT INTO tableout SELECT IFNULL(nomlabo, \"CICG\"), count(*)
flotout, sum(taille) octetout FROM flotsrecents F LEFT JOIN ipreseau R ON R.ipsrc1 = F.ipsrc1
and R.ipsrc2 = F.ipsrc2 and R.ipsrc3 = F.ipsrc3 GROUP BY R.nomlabo");
        if (mysql_query(db_thread, inter) != 0)
      {
            log = fopen(LOG, "a");
                fprintf(log, "%s : process_query() insertion into tableout failed\n",
date_new);
                fclose(log);
                printf("process_query() insertion into tableout failed\n");
        }
```

```
        sprintf(inter, "INSERT INTO tableoutbis SELECT distinct(I.nomlabo),flotout, octetout
FROM ipreseau I LEFT JOIN tableout T ON I.nomlabo=T.nomlabo");

        if (mysql_query(db_thread, inter) != 0)
      {
            log = fopen(LOG, "a");
                fprintf(log, "%s : process_query() insertion into tableoutbis failed\n",
date_new);
                fclose(log);
                printf("process_query() insertion into tableoutbis failed\n");
        }


/*(S)*/
      /*-- on interroge le routeur pour connaitre le nombre d'octets en entree et en
      sortie sur tous les ports */

      sprintf(chaine, "snmptable %s public interfaces.ifTable", adr_routeur);
      num = tube_ouvrir(chaine, "r");

      /*-- dans la reponse il y a 3 lignes pourries qu'on recupere et qu'on ecrase
      tout de suite */

      strcpy(chaine, "");
      i = 0;
      while (read(num, &car, 1) > 0 && car != '\n')
      {
            chaine[i] = car;
            i++;
      }
      strcpy(chaine, "");
      i = 0;
      while (read(num, &car, 1) > 0 && car != '\n')
      {
            chaine[i] = car;
            i ++;
      }
      strcpy(chaine, "");
      i = 0;
      while (read(num, &car, 1) > 0 && car != '\n')
      {
            chaine[i] = car;
            i ++;
      }
      chaine[i] = '\0';

      /*-- pour chaque port du routeur : */
      while (read(num, &car, 1) > 0)
      {
            /* on recupere la ligne */
            strcpy(chaine, "");
            out = 0;
            in = 0;
            num_port = 0;
            i = 0;
            while (read(num, &car, 1) > 0 && car != '\n')
            {
                  chaine[i] = car;
                  i ++;
            }
            chaine[i] = '\0';

            /*-- on parse la ligne pour recuperer les 3 infos qui nous interesse */
            sscanf(chaine, "%*s %*d %*d %*d %*d %*d %u %*d %*d %*d %*d %*d %u %*s %*s %*s %*s
%*d %*d %*s %*s %d", &in, &out, &num_port);
            if (num_port == 0)
                  sscanf(chaine, "%*s %*d %*d %*d %*d %*d %u %*d %*d %*d %*d %*d %u %*s %*s
%*s %*d %*d %*s %*s %d", &in, &out, &num_port);

            /*-- on calcule le nombre d'octets ayant circules sur l'interface
            depuis la fois d'avant (0 si c'est la premiere fois) */
            /* valeur max d'un compteur SNMP : 4294967295 */
            if (!dump1)
            {
                  if ((deltain = in - old[num_port - 1].in) < 0)
                        deltain = (MAXSNMP - old[num_port - 1].in) + in;
```

```
                  if ((deltaout = out - old[num_port - 1].out) < 0)
                        deltaout = (MAXSNMP - old[num_port - 1].out) + out;
            }
            else
            {
                  deltain = 0;
                  deltaout = 0;
            }

            /*-- on affecte les valeurs au labo correspondant au numero de port */
            sprintf(inter, "INSERT INTO snmp SELECT nomlabo, %u, %u FROM laboratoires WHERE
port = %d", 8*deltain, 8*deltaout, num_port);
            if (mysql_query(db_thread, inter) != 0)
                {
                        log = fopen(LOG, "a");
                        fprintf(log, "%s : process_query() update into valeurs_labo failed\n",
date_new);
                        fclose(log);
                        printf("process_query() insert into snmp failed\n");
                }

            //printf("%d IN : %u / %u, OUT : %u / %u\n", num_port, old[num_port - 1].in, in,
old[num_port - 1].out, out);

            /*-- on stocke la nouvelle valeur */
            old[num_port - 1].in = in;
            old[num_port - 1].out = out;
      }
      tube_fermer(num);

/*(T)*/
      /*-- on reunit toutes les valeurs dans la meme table
      (valeurs de sortie, valeurs d'entree et interrogation snmp) */

        sprintf(inter, "INSERT INTO valeurs_labo SELECT T1.nomlabo,
\"%s\",ROUND(IFNULL(octetin,0)/%d,2), ROUND(IFNULL(octetout,0)/%d,2),
ROUND(IFNULL(flotin,0)/%d,2), ROUND(IFNULL(flotout, 0)/%d,2), ROUND(snmpin/%d,2),
ROUND(snmpout/%d,2) FROM tableinbis T1, tableoutbis T2, snmp S WHERE T1.nomlabo = T2.nomlabo
AND T2.nomlabo = S.nomlabo", date_new, secd, secd, secd, secd, secd, secd);
        if (mysql_query(db_thread, inter) != 0)
      {
            log = fopen(LOG, "a");
                fprintf(log, "%s : process_query() insertion into valeurs_labo failed\n",
date_new);
                fclose(log);
                printf("process_query() insertion into valeurs_labo failed\n");
                //return -1;
        }

/*(U)*/
      /*-- on supprime toutes les tables temporaires */
        mysql_query(db_thread, "DELETE FROM flotsrecents");
        mysql_query(db_thread, "DELETE FROM tablein");
        mysql_query(db_thread, "DELETE FROM tableout");
        mysql_query(db_thread, "DELETE FROM tableinbis");
        mysql_query(db_thread, "DELETE FROM tableoutbis");
      mysql_query(db_thread, "DELETE FROM snmp");

      /*-- on supprime la table flots tous les week-end a cause du risque de depassement de
capacite */
      jour_semaine = infos->tm_wday;      // 0 -> dimanche
      if (dump1) jour_semaine_old = jour_semaine;
      /*-- on efface la table quand on passe de samedi a dimanche*/
      if (jour_semaine == 0 && jour_semaine_old == 6)
      {
            sprintf(inter, "DROP TABLE %s", table);
            mysql_query(db_thread, inter);
      }
      jour_semaine_old = jour_semaine;

        fermer_base_thread();
      if (dump1) dump1 = 0;
//        printf("JE SUIS PARTI !!!!\n");

/*(V)*/
      /*-- on debloque le verrou */
      pthread_mutex_unlock (&mutex);
```

```
      nb_attente --;
      pthread_exit (0);
}


/*(J)*/

/*****************************************************************************************
* on ajoute un flot a la table soit quand le flux est termine (TCP) soit parce qu'on dumpe *
* (TCP et UDP) soit parce qu'on a recu un paquet (ICMP).                                    *
*****************************************************************************************/

int remplir_base(flux elt)
{
        char tmp[50];
        char inter[1000];
        char adr1s[4], adr2s[4], adr3s[4], adr4s[4], adr1d[4], adr2d[4], adr3d[4], adr4d[4];
        int i, j, point, old_j;

//        printf("on insere :");
//        printf("deb=%s fin=%s @src=%s psrc=%d @dest=%s pdest=%d oct=%d\n", elt.heureDebut,
elt.heureFin, elt.adrSrc, elt.portSrc, elt.adrDest, elt.portDest, elt.nbOct);

      /*-- on decompose les 2 adresses ip en 4 champs chacune */
        point = 0;
        j = 0;
        for (i=0; elt.adrSrc[i] != '\0';i++)
        {
                if (elt.adrSrc[i] == '.')
                {
                        point += 1;
                        old_j = j;
                        j = 0;
                } else
                {
                        if (point == 0) adr1s[j] = elt.adrSrc[i];
                        else if (point == 1)
                        {
                                adr2s[j] = elt.adrSrc[i];
                                adr1s[old_j] = '\0';
                        }
                        else if (point == 2)
                        {
                                adr3s[j] = elt.adrSrc[i];
                                adr2s[old_j] = '\0';
                        }
                        else if (point == 3)
                        {
                                adr4s[j] = elt.adrSrc[i];
                                adr3s[old_j] = '\0';
                        }
                        j++;
                }
        }
        adr4s[j] = '\0';
        point = 0;
        j = 0;
        for (i=0; elt.adrDest[i] != '\0';i++)
      {
                if (elt.adrDest[i] == '.')
            {
                        point += 1;
                        old_j = j;
                        j = 0;
                } else
            {
                        if (point == 0) adr1d[j] = elt.adrDest[i];
                        else if (point == 1)
                        {
                                adr2d[j] = elt.adrDest[i];
                                adr1d[old_j] = '\0';
                        }
                        else if (point == 2)
                        {
                                adr3d[j] = elt.adrDest[i];
                                adr2d[old_j] = '\0';
                        }
```

```
                        else if (point == 3)
                        {
                                adr4d[j] = elt.adrDest[i];
                                adr3d[old_j] = '\0';
                        }
                        j++;
                }
        }
        adr4d[j] = '\0';

      /*-- on insere le flux dans la table *//*CHAUSS*/
        sprintf(inter, "insert into %s values(NULL, %s, %s, %s, %s, %s, %s, %s, %s, %d, \"%s
%s\", \"%s %s\", %d, %d)", table, adr1s, adr2s, adr3s, adr4s, adr1d, adr2d, adr3d, adr4d,
elt.portDest, tdate, elt.heureDebut, tdate, elt.heureFin, 8*elt.nbOct, elt.proto);
//        printf("%s\n", inter);
        if (mysql_query(db, inter) != 0)
      {
            log = fopen(LOG, "a");
                fprintf(log, "%s : process_query() insertion into %s failed\n", date_new,
table);
                fclose(log);
                printf("process_query() failed\n");
                return -1;
        }
        return 0;
}

/*(M)*/
/*****************************************************************************
* supprime un flux TCP ou UDP termine de la liste et appelle remplir_base (flag F  *
* ou R ou dump).                                                                    *
*****************************************************************************/

void terminer_flux(flux elt)
{
        l_flux *courant, *prec;
      if (elt.proto == 6)
      {
            courant = teteTCP;
            prec = teteTCP;
      } else
      {
            courant = teteUDP;
                prec = teteUDP;
      }
        while (!(  courant == NULL
               || (!strcmp(courant->element.adrSrc, elt.adrSrc)
               && (courant->element.portSrc == elt.portSrc)
               && !strcmp(courant->element.adrDest, elt.adrDest)
               && (courant->element.portDest == elt.portDest))))
      {
                prec = courant;
                courant = courant->suivant;
        }
        if (courant != NULL)
      {
                strcpy(courant->element.heureFin, elt.heureFin);
                courant->element.nbOct += elt.nbOct;
                remplir_base(courant->element);
                if (teteTCP == courant)
            {
                        teteTCP = teteTCP->suivant;
                        courant = NULL;
                } else if (teteUDP == courant)
            {
                  teteUDP = teteUDP->suivant;
                  courant = NULL;
            } else
            {
                        prec->suivant = courant->suivant;
                }
        } else
      {

                remplir_base(elt);
        }
        free(courant);
```

```
        prec = NULL;
        free(prec);
}


/*(N)*/
/*****************************************************************************
* dump le contenu des 2 listes dans la table cad enleve toutes les connexions en   *
* cours de la liste (termine tous les flux)                                        *
*****************************************************************************/

void dumperListe()
{
        char time_tmp[20];
        while (teteTCP != NULL)
      {
                sprintf(time_tmp,"%d", hour);
                strcpy(teteTCP->element.heureFin, time_tmp);
                strcat(teteTCP->element.heureFin, ":");
                sprintf(time_tmp,"%d", min_new);
                strcat(teteTCP->element.heureFin, time_tmp);
                strcat(teteTCP->element.heureFin, ":");
                sprintf(time_tmp,"%d", sec);
                strcat(teteTCP->element.heureFin, time_tmp);
                remplir_base(teteTCP->element);
                teteTCP = teteTCP->suivant;
        }
        while (teteUDP != NULL)
      {
                sprintf(time_tmp,"%d", hour);
                strcpy(teteUDP->element.heureFin, time_tmp);
                strcat(teteUDP->element.heureFin, ":");
                sprintf(time_tmp,"%d", min_new);
                strcat(teteUDP->element.heureFin, time_tmp);
                strcat(teteUDP->element.heureFin, ":");
                sprintf(time_tmp,"%d", sec);
                strcat(teteUDP->element.heureFin, time_tmp);
                remplir_base(teteUDP->element);
                teteUDP = teteUDP->suivant;
        }
}

/*(E')*/

/*****************************************************************************
* routine appelee par pcap_loop a chaque reception de paquet                  *
*****************************************************************************/

void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet);

void got_packet(u_char *args, const struct pcap_pkthdr *header, const u_char *packet)
{
/*(F)*/
        /*-- definit le contenu du paquet*/
        const struct sniff_ethernet *ethernet;  /* ethernet header */
        const struct sniff_ip *ip;              /* IP header */
        const struct sniff_tcp *tcp;            /* TCP header */

        /*-- definit les tailles des structures utilisees */
        int size_ethernet = sizeof(struct sniff_ethernet);
        int size_ip = sizeof(struct sniff_ip);
        int size_tcp = sizeof(struct sniff_tcp);

      register int s;

      int proto, portSrc, portDest, nbOct, flag, i, pid;
      char date[9], adrSrc[16], adrDest[16];

      char time_tmp[20];
      int nbsec_new, nbsec_old;
      flux tmp;

      pthread_attr_t thread_attr;
      pthread_t th1;


        /*-- initialise les structures */
```

```
        ethernet = (struct sniff_ethernet*)(packet);
        ip = (struct sniff_ip*)(packet + size_ethernet);
        tcp = (struct sniff_tcp*)(packet + size_ethernet + size_ip);

/*(G)*/
      /*-- calcul de la date */
      temps = time(NULL);
        infos = localtime(&temps);
        i = 1900 + infos->tm_year;
        sprintf(time_tmp,"%d", i);
        strncpy(tdate, time_tmp, 5);
        strcat(tdate, "-");
        i = 1 + infos->tm_mon;
        sprintf(time_tmp,"%d", i);
        strcat(tdate, time_tmp);
        strcat(tdate, "-");
        sprintf(time_tmp,"%d", infos->tm_mday);
        strcat(tdate, time_tmp);

/*(H)*/
      /*-- decode les differentes infos dont on a besoin */
      proto = ip->ip_p;

      hour = infos->tm_hour;
      min = infos->tm_min;
      sec = infos->tm_sec;
      sprintf(date, "%02d:%02d:%02d", hour, min, sec);

      strcpy(adrSrc, inet_ntoa(ip->ip_src));
      portSrc = ntohs(tcp->th_sport);
      strcpy(adrDest, inet_ntoa(ip->ip_dst));
      portDest = ntohs(tcp->th_dport);
      flag = tcp->th_flags;
      nbOct = (int)ntohs(ip->ip_len);

/*(I)*/
      /*-- selon le protocole du paquet, on lance differentes operations*/
      switch (ip->ip_p)
      {

      /*-- si le protocole est icmp, un paquet = 1 flux, donc on insere dans la base
         des qu'on a un paquet */
/*(1)*/     case 1:
            tmp.proto = 1;
            strcpy(tmp.heureDebut, date);
                strcpy(tmp.heureFin, date);
                strcpy(tmp.adrSrc, adrSrc);
                tmp.portSrc = 0;
                strcpy(tmp.adrDest, adrDest);
                tmp.portDest = 0;
                tmp.nbOct = nbOct;
                //connecter_base();
            remplir_base(tmp);
                //fermer_base();
            break;

      /*-- pour tcp, un flux debute par un flag S et se termine par un flag F ou R
         (sauf certaines connexions en particulier http ne sont pas terminees normalement)
         on regarde donc les flags du paquets*/
/*(6)*/     case 6:
            tmp.proto = 6;
            strcpy(tmp.heureDebut, date);
                strcpy(tmp.adrSrc, adrSrc);
                tmp.portSrc = portSrc;
                strcpy(tmp.adrDest, adrDest);
                tmp.portDest = portDest;

            /*-- si on a un flag S on cree un nouveau flux dans la liste et on attend le
               paquet suivant*/
            if (flag & TH_SYN)
            {
                        strcpy(tmp.heureFin, "en_cours");
                        tmp.nbOct = 0;
                        inserer1flux(tmp);
            }

            /*-- si on a un flag F on rajoute une ligne a la table et on enleve ce flot
```

```
               de la liste */
            else if (flag & TH_FIN)
            {
                        strcpy(tmp.heureFin, date);
                        tmp.nbOct = nbOct;
                        //connecter_base();
                        terminer_flux(tmp);
                        //fermer_base();
            }

            /*-- si on a un flag Rst on fait la meme chose qu'avec le flag F et on fait de
               meme avec le flux inverse.*/
            else if (flag & TH_RST)
            {
                        strcpy(tmp.heureFin, date);
                        tmp.nbOct = nbOct;
                        //connecter_base();
                        terminer_flux(tmp);
                        //fermer_base();
                        strcpy(tmp.adrSrc, adrDest);
                        tmp.portSrc = portDest;
                        strcpy(tmp.adrDest, adrSrc);
                        tmp.portDest = portSrc;
                        //connecter_base();
                        terminer_flux(tmp);
                        //fermer_base();
            }

            /*-- on se contente de rechercher ce flux dans la liste et on incremente sa
                 taille s'il existe deja et on le cree sinon*/
            else
            {
                        strcpy(tmp.heureFin, "en_cours");
                        tmp.nbOct = nbOct;
                        mettre_a_jour_nbOct(tmp);
            }
            break;
/*(17)*/case 17:

            /*-- dans udp, on recherche si le paquet appartient a un flux deja existant
               s'il y est on incremente sa taille sinon on le cree*/
            tmp.proto = 17;
            strcpy(tmp.heureDebut, date);
                strcpy(tmp.heureFin, "en_cours");
                strcpy(tmp.adrSrc, adrSrc);
                tmp.portSrc = portSrc;
                strcpy(tmp.adrDest, adrDest);
                tmp.portDest = portDest;
                tmp.nbOct = nbOct;
            mettre_a_jour_nbOct(tmp);
            break;
      default:
            break;
      }

      /*-- apres le traitement du paquet on verifie s'il est necessaire de faire un nouveau
         dump*/
      sscanf(date, "%d:%d:%d", &hour_new, &min_new, &sec_new);
      nbsec_new = min_new * 60 + sec_new;
      nbsec_old = min_old * 60 +sec_old;
      if (((nbsec_new > nbsec_old) && (nbsec_new - nbsec_old >= secd)) || ((nbsec_new <
nbsec_old) && (abs(nbsec_new - nbsec_old)  <= 3600 - secd)))
        {
//                printf("JE DUMPE !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n");
               //connecter_base();
               dumperListe();
               //fermer_base();

            sprintf(date_new, "%s %d:%d:%d",tdate, hour_new, min_new, sec_new);
            sprintf(date_old, "%s %d:%d:%d",tdate, hour_old, min_old, sec_old);

            /*-- on cree un nouveau thread */
            if (pthread_attr_init (&thread_attr) != 0) {
                  fprintf (stderr, "pthread_attr_init error");
                  exit (1);
            }
```

```
            if (pthread_attr_setdetachstate (&thread_attr, PTHREAD_CREATE_DETACHED) != 0) {
                  fprintf (stderr, "pthread_attr_setdetachstate error");
                  exit (1);
            }

                if (pthread_create (&th1, &thread_attr, constr_table_valeur, NULL) < 0) {
                  fprintf (stderr, "pthread_create error for thread 1\n");
                  exit (1);
            }

                hour_old = hour_new;
                min_old = min_new;
                sec_old = sec_new;
        }

}


/*************************************************************************
* permet de recuperer le filtre a appliquer depuis la ligne de commande *
*************************************************************************/

char *
copy_argv(register char **argv)
{
        register char **p;
        register u_int len = 0;
        char *buf;
        char *src, *dst;

        p = argv;
        if (*p == 0)
                return 0;

        while (*p)
                len += strlen(*p++) + 1;

        buf = (char *)malloc(len);
        if (buf == NULL)
                error("copy_argv: malloc");

        p = argv;
        dst = buf;
        while ((src = *p++) != NULL) {
                while ((*dst++ = *src++) != '\0')
                        ;
                dst[-1] = ' ';
        }
        dst[-1] = '\0';

        return buf;
}


/*************************************************************************
* programme principal                                                    *
*************************************************************************/

int main(int argc, char **argv)
{
      register int op;
      extern char *optarg;
      extern int optind;
      char *filter;
        char *dev;                      /* interface sniffee */
        char errbuf[PCAP_ERRBUF_SIZE];  /* chaine d'erreur */
        pcap_t *descr;                  /* Sniff handler */

        struct bpf_program fp;          /* hold compiled program */
        bpf_u_int32 maskp;              /* subnet mask */
        bpf_u_int32 netp;               /* ip */
        char filter_app[50] = "";
      int numOfPackets = 0;         /* on sniff tant qu'il n'y a pas d'erreur */
      int i;
      char time_tmp[20];

      /*-- on initialise le verrou */
```

```
      pthread_mutex_init (&mutex, NULL);

      /*-- on recupere les options de la ligne de commande */
      filter = NULL;
      dev = NULL;


/*(A)*/
      /*-- calcul de l'heure initiale */
      temps = time(NULL);
        infos = localtime(&temps);
        i = 1900 + infos->tm_year;
        sprintf(time_tmp,"%d", i);
        strncpy(tdate, time_tmp, 5);
        strcat(tdate, "-");
        i = 1 + infos->tm_mon;
        sprintf(time_tmp,"%d", i);
        strcat(tdate, time_tmp);
        strcat(tdate, "-");
        sprintf(time_tmp,"%d", infos->tm_mday);
        strcat(tdate, time_tmp);

      min_old = infos->tm_min;
        sec_old = infos->tm_sec;
        hour_old = infos->tm_hour;
      sprintf(date_old, "%s %d:%d:%d",tdate, hour_old, min_old, sec_old);

/*(B)*/
      /*-- on entre la liste des options possibles et on ajoute un : quand elles
      ont un argument */

      while ((op = getopt(argc, argv, "i:t:hn:r:")) != -1)
      {
            switch (op) {
            case 'i' :
                  dev = optarg;
                  break;
            case 't' :
                  strcpy(table, optarg);
                  break;
            case 'h' :
                  (void)fprintf(stderr, "Usage:  [-t table] [ -i interface ] [ -n secondes ] [
-r routeur ] [ expression ] \n");
                  (void)fprintf(stderr, "\t\ttable : table de la base netflow dans laquelle on
insere les flots (defaut:flots)\n");
                  (void)fprintf(stderr, "\t\tinterface : interface a sniffer (defaut :
eth0)\n");
                  (void)fprintf(stderr, "\t\trouteur : adresse ip du routeur a interroger (par
defaut 147.173.1.103)\n");
                  (void)fprintf(stderr, "\t\tsecondes : intervalle de temps entre chaque dump
(defaut : 60)\n");
                  (void)fprintf(stderr, "\t\texpression : filtre a appliquer au sniff (defaut
: -)\n");
                  exit(-1);
            case 'n' :
                  if (atoi(optarg) > 3600) {
                        (void)fprintf(stderr, "L'intervalle entre chaque dump doit etre
inferieur a 3600s\n");
                        exit(-1);
                  }
                  secd = atoi(optarg);
                  break;
            case 'r' :
                  strcpy(adr_routeur, optarg);
                  break;
            default : break;
            }
      }
      filter = copy_argv(&argv[optind]);
      if (filter != NULL) {
            strcpy(filter_app, filter);
      }

      /*-- on recupere la table qu'on veut remplir (par defaut flots) et
      on initialise les listes*/

      teteTCP = NULL;
```

```
      teteUDP = NULL;


        /*-- choix de l'interface */
      if (dev == NULL)
      {
            dev = pcap_lookupdev(errbuf);
            pcap_lookupnet(dev, &netp, &maskp, errbuf);
      }

        /*-- affichage de la configuration */
      printf("Table: [%s]\n", table);
        printf("Interface: [%s]\n", dev);
        printf("Routeur: [%s]\n", adr_routeur);
        printf("Intervalle: [%d]\n", secd);
        printf("Filtre: [%s]\n", filter_app);

/*(C)*/
        /*-- ouverture de l'interface au sniff */
        descr = pcap_open_live(dev, BUFSIZ, 1, 0, errbuf);
        if (descr == NULL)
        {
            log = fopen(LOG, "a");
                fprintf(log, "%s : pcap_open_live(): %s\n", date_old, errbuf);
                fclose(log);
                printf("pcap_open_live(): %s\n", errbuf);
                exit(1);
        }

/*(D)*/
        /*-- application du filtre (par defaut rien) */
        if( pcap_compile(descr, &fp, filter_app, 0, netp) == -1)
        {
            log = fopen(LOG, "a");
                fprintf(log, "%s : pcap_compile : echec\n", date_old);
                fclose(log);
                printf("pcap_compile : echec\n");
                exit(1);
        }
        if (pcap_setfilter(descr, &fp) == -1)
        {
            log = fopen(LOG, "a");
                fprintf(log, "%s : pcap_setfilter : echec\n", date_old);
                fclose(log);
                printf("pcap_setfilter : echec\n");
                exit(1);
        }

      dump1 = 1;

      /*-- ouverture de la base de donnees*/
      connecter_base();

/*(E)*/
        /*-- lancement de la boucle de capture
      pcap_loop appelera got_packet a chaque fois qu'il recevra un paquet*/

        pcap_loop(descr, numOfPackets, got_packet, NULL);
        pcap_close(descr);

      fermer_base();

        printf("End\n");
      return(0);
}
```

Base de données

I. Structure des tables

A. Table FLOTS:

1. Creation de la table

```
CREATE TABLE flots (
  numflot          bigint(20) unsigned not null auto_increment,
  ipsrc1           tinyint(3) unsigned,
  ipsrc2           tinyint(3) unsigned,
  ipsrc3           tinyint(3) unsigned,
  ipsrc4           tinyint(3) unsigned,
  ipdst1           tinyint(3) unsigned,
  ipdst2           tinyint(3) unsigned,
  ipdst3           tinyint(3) unsigned,
  ipdst4           tinyint(3) unsigned,
  portdst          mediumint(8) unsigned not null,
  starttime        datetime,
  endtime          datetime,
  taille           bigint(20) unsigned not null,
  typprot          tinyint(3) unsigned not null,
  primary key(numflot));
```

2. Contenu de la table

Nom de la colonne	Description
numflot	numéro d'identifiant d'un flot
ipsrc1	premier octet de l'adresse ip source
ipsrc2	deuxième octet de l'adresse ip source
ipsrc3	troisième octet de l'adresse ip source
ipsrc4	quatrième octet de l'adresse ip source
ipdst1	premier octet de l'adresse ip destination
ipdst2	deuxième octet de l'adresse ip destination
ipdst3	troisième octet de l'adresse ip destination
ipdst4	quatrième octet de l'adresse ip destination
portdst	numéro du port destination
starttime	date et heure de début du transfert
endtime	date et heure de fin du transfert
taille	nombre d'octets transferes
typprot	numéro du type de protocole

B. Table IPRESEAU:

1. Creation de la table

```
CREATE TABLE ipreseau (
  ipsrc1     tinyint(3) unsigned not null,
  ipsrc2     tinyint(3) unsigned not null,
  ipsrc3     tinyint(3) unsigned not null,
  ipsrc4     tinyint(3) unsigned not null,
  nomlabo    varchar(25) not null,
  primary key(ipsrc1, ipsrc2, ipsrc3, ipsrc4));
```

2. Contenu de la table

Nom de la colonne	Description
ipsrc1	Premier octet de l'adresse ip
ipsrc2	deuxième octet de l'adresse ip
ipsrc3	troisieme octet de l'adresse ip
ipsrc4	quatrieme octet de l'adresse ip
nomlabo	nom du laboratoire

C. Table VALEURS_LABO:

1. Creation de la table

```
CREATE TABLE valeurs_labo (
  nomlabo    varchar(25) not null,
  timestamp  datetime not null,
  octetin    bigint(20) unsigned not null,
  octetout   bigint(20) unsigned not null,
  flotin     bigint(20) unsigned not null,
  flotout    bigint(20) unsigned not null,
  snmpin     bigint(20) unsigned not null,
  snmpout    bigint(20) unsigned not null,
  primary key(nomlabo, timestamp));
```

2. Contenu de la table

Nom de la colonne	Description
nomlabo	nom du laboratoire
timestamp	date et heure du dump
octetin	nombre d'octest entrants « utiles »
octetout	nombre d'octets sortants « utiles »
flotin	nombre de flux entrants « utiles »
flotout	nombre de flux sortants « utiles »
snmpin	nombre d'octets entrants total
snmpout	nombre d'octets sortants total

D. Table LABORATOIRES:

1. Creation de la table

```
CREATE TABLE laboratoires (
  nomlabo  varchar(25) not null,
  adrlabo  varchar(100) not null,
  domrech  varchar(255) not null,
  teleph   varchar(100) not null,
  fax      varchar(100) not null,
  email         varchar(100) not null,
  debit         bigint(20) unsigned not null,
  NbFlotMax     bigint(20) unsigned not null,
  port          tinyint(3) unsigned not null,
  primary key(nomlabo));
```

2. Contenu de la table

Nom de la colonne	Description
nomlabo	nom du laboratoire
adrlabo	adresse du laboratoire
domrech	dommaine de recherchedu laboratoire
teleph	téléphone du laboratoire
fax	fax du laboratoire
email	email du laboratoire
debit	débit du brins du laboratoire
NbFlotMax	nombre de flot maximum du laboratoire
port	Port du routeur sur lequel est branché le laboratoire

E. Table ALERTE:

1. Creation de la table

```
CREATE TABLE valeurs_labo (
  nomlabo    varchar(25) not null,
  timestamp  datetime not null,
  octetin    bigint(20) unsigned not null,
  octetout   bigint(20) unsigned not null,
  flotin     bigint(20) unsigned not null,
  flotout    bigint(20) unsigned not null,
  snmpin     bigint(20) unsigned not null,
  snmpout    bigint(20) unsigned not null,
  primary key(nomlabo, timestamp));
```

2. Contenu de la table

Nom de la colonne	Description
nomlabo	nom du laboratoire
timestamp	date et heure du dump
octetin	nombre d'octest entrants « utiles »
octetout	nombre d'octets sortants « utiles »
flotin	nombre de flux entrants « utiles »
flotout	nombre de flux sortants « utiles »
snmpin	nombre d'octets entrants total
snmpout	nombre d'octets sortants total

F. Tabble PROTOCOLE :

1. Creation de la table

```
CREATE TABLE protocole (
  numport   mediumint(8) unsigned not null,
  nomproto  varchar(6),
  nomappli  varchar(8),
  primary key(numport));
```

2. Contenu de la table

Nom de la colonne	Description
numport	numero du port
nomproto	type de protocole
nomappli	nom de l'application

G. Tabble SNMP:

1. Creation de la table

```
CREATE TABLE snmp (
  nomlabo   varchar(25) not null,
  snmpin    bigint(20),
  snmpout   bigint(20));
```

2. Contenu de la table

Nom de la colonne	Description
nomlabo	nom du laboratoire
snmpin	nombre d'octets entrant total
snmpout	nombre d'octets sortant total

H. Table FLOTSRECENT:

1. Creation de la table

```
CREATE TABLE flotsrecent (
  numflot   bigint(20) unsigned not null auto_increment,
  ipsrc1    tinyint(3) unsigned not null,
  ipsrc2    tinyint(3) unsigned not null,
  ipsrc3    tinyint(3) unsigned not null,
  ipsrc4    tinyint(3) unsigned not null,
  ipdest1   tinyint(3) unsigned not null,
  ipdest2   tinyint(3) unsigned not null,
  ipdest3   tinyint(3) unsigned not null,
  ipdest4   tinyint(3) unsigned not null,
  taille    bigint(20) unsigned not null,
  primary key(numflot));
```

I. Table TABLEIN:

1. Creation de la table

```
CREATE TABLE tablein (
  nomlabo   varchar(25) not null,
  flotin    bigint unsigned not null,
  octetin   bigint unsigned not null,
  primary key(nomlabo));
```

J. Table TABLEOUT:

1. Creation de la table

```
 CREATE TABLE tableout (
   nomlabo  varchar(25) not null,
   flotout  bigint unsigned not null,
   octetout bigint unsigned not null,
   primary key(nomlabo));
```

K. Table TABLEINBIS:

1. Creation de la table

```
CREATE TABLE tableinbis (
  nomlabo   varchar(25) not null,
  flotin    bigint unsigned not null,
  octetin   bigint unsigned not null,
  primary key(nomlabo));
```

L. Table TABLEOUTBIS:

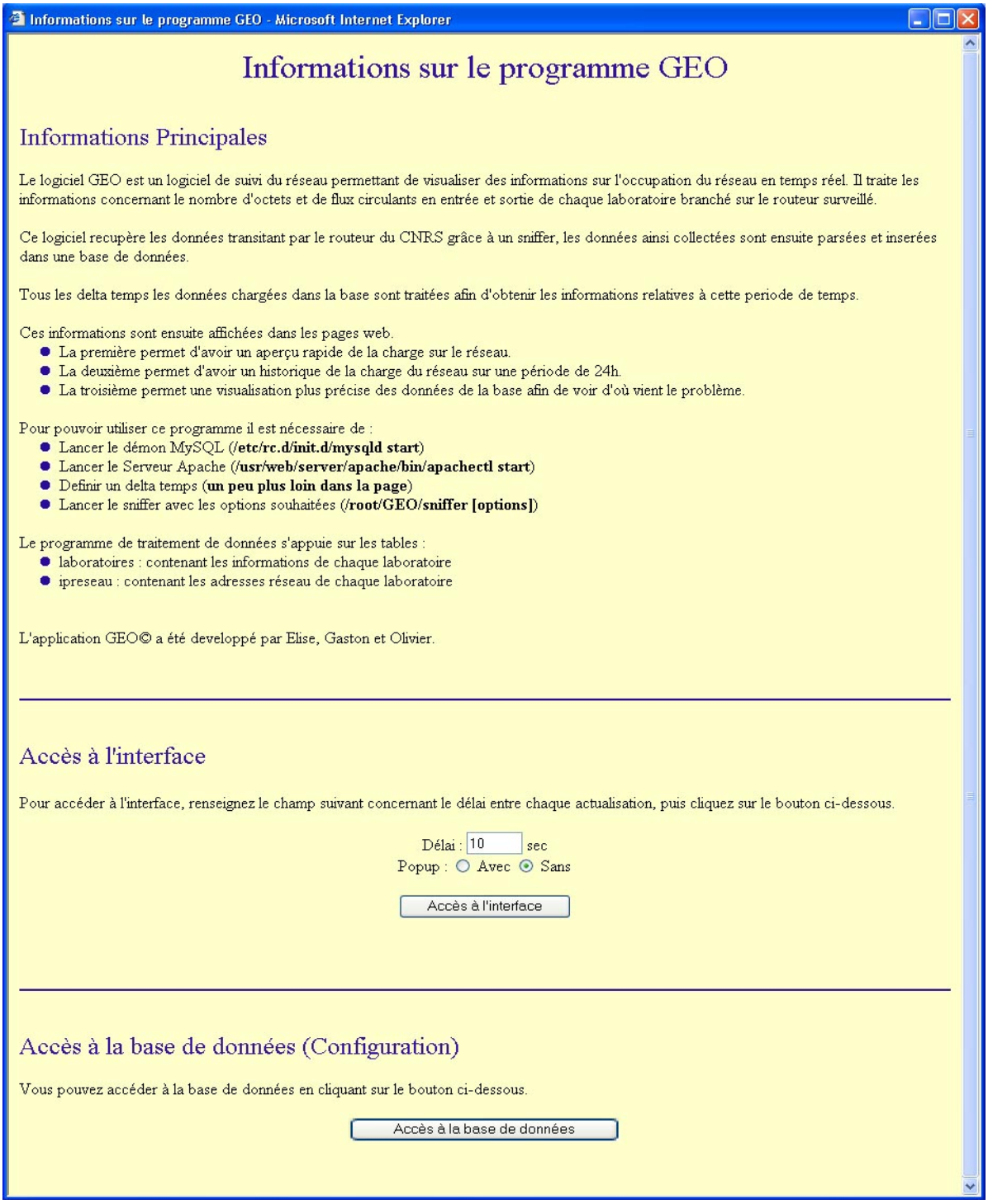
1. Creation de la table

```
CREATE TABLE tableoutbis (
  nomlabo  varchar(25) not null,
  flotout  bigint unsigned not null,
  octetout bigint unsigned not null,
  primary key(nomlabo));
```

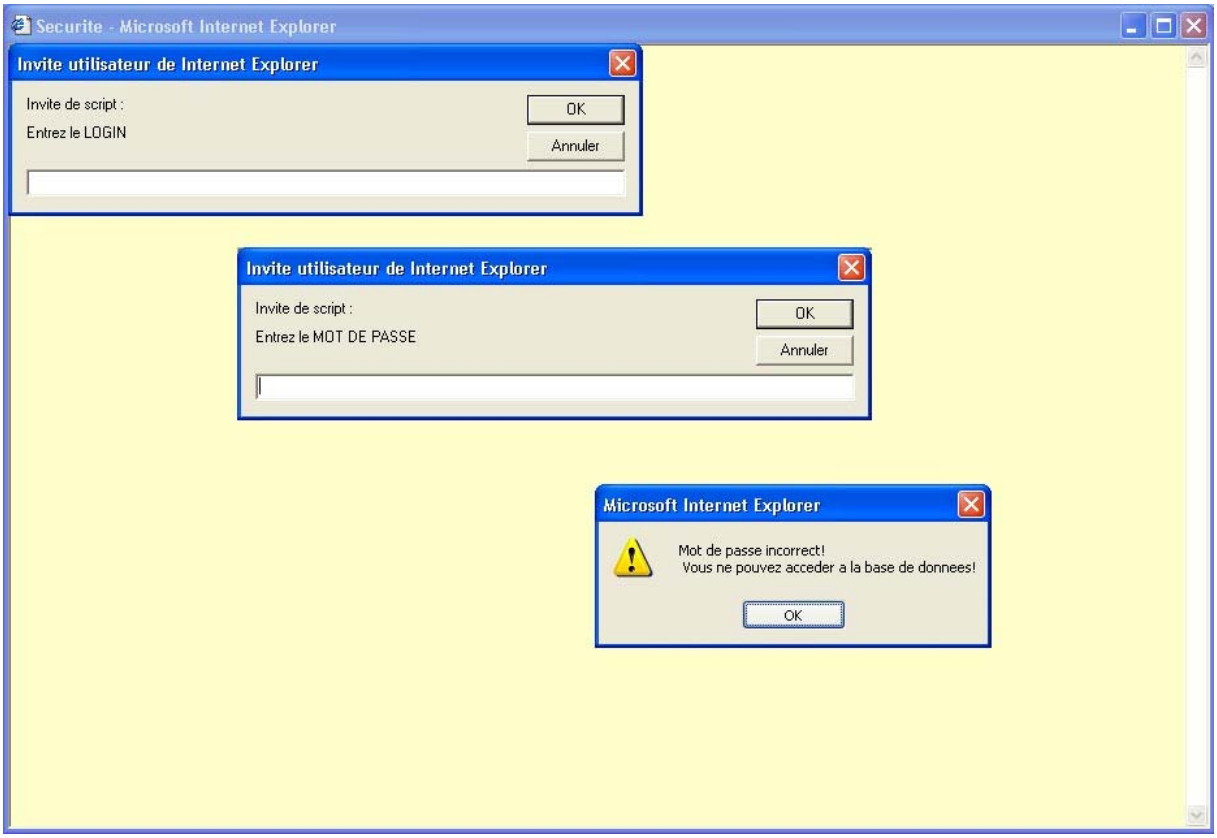
Interface

I. Information

A. Information.php



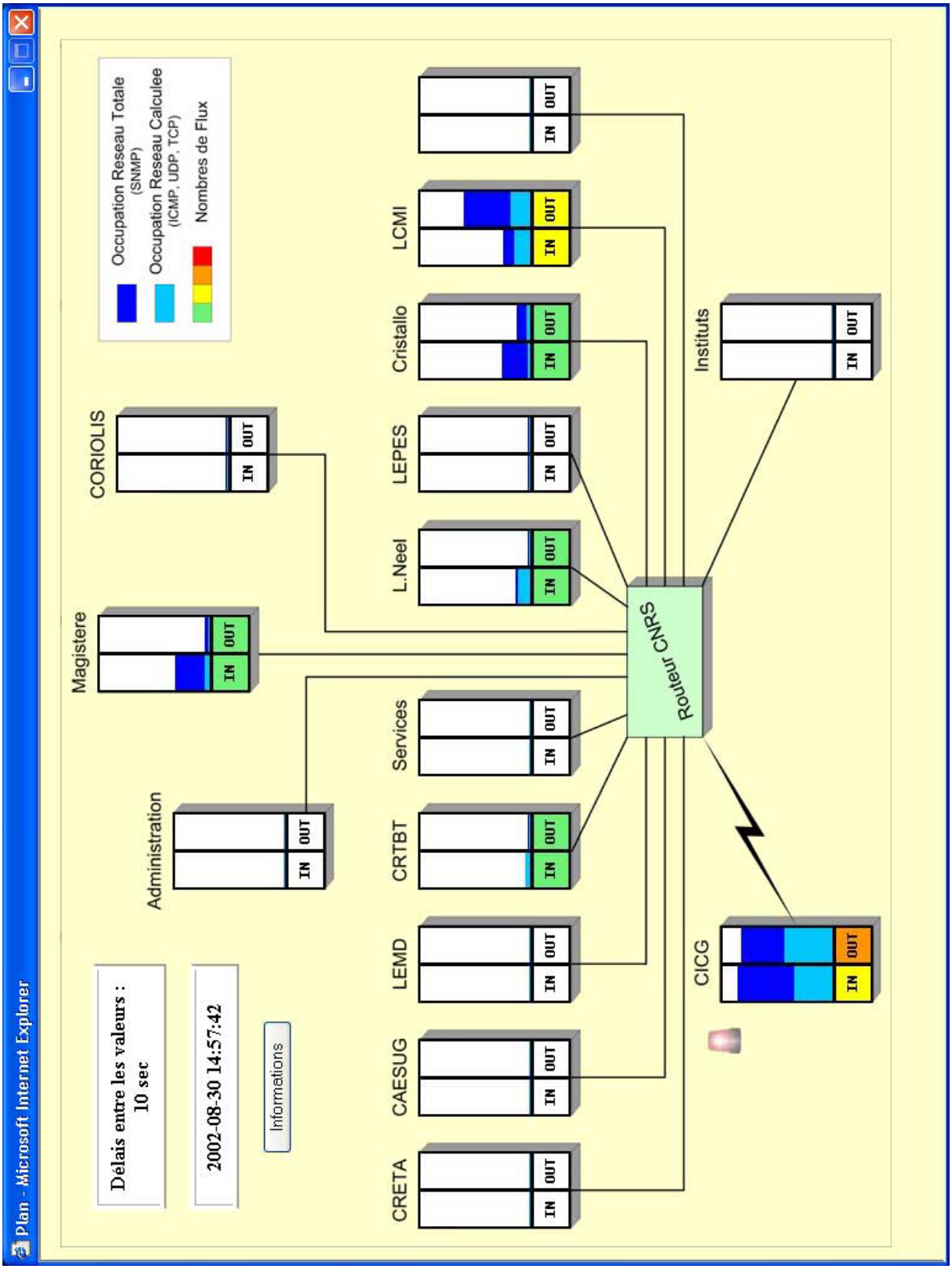
B. Securite.html



II. Plan

A. Plan.php

1. Copie d'écran



2. Code

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE="JavaScript">
    window.moveTo(0,0);
    if (document.all)
    {
      window.resizeTo(screen.availWidth,screen.availHeight);
    }
    else if (document.layers)
    {
      if (window.outerHeight<screen.availHeight||window.outerWidth<screen.availWidth)
      {
        window.outerHeight = screen.availHeight;
        window.outerWidth = screen.availWidth;
      }
    }
  </SCRIPT>

  <TITLE>Plan</TITLE>

  <META HTTP-EQUIV="Refresh" CONTENT="<?php echo $delta_tps ?>">
  <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">
</HEAD>

<BODY BGCOLOR=#FFFFCC>

  <SCRIPT>
    function OuvPage(x,nomPage,option)
    {
      nouvFen = window.open(x,nomPage,option);
      nouvFen.focus();
    }

    function FermPage(x,nomPage,option)
    {
      nouvFen = window.open(x,nomPage,option);
      nouvFen.close();
    }
  </SCRIPT>

  <?php
    $user="netflow";
    $host="localhost";
    $password="netflow";
    $database="netflow";

    mysql_connect($host, $user, $password);
    mysql_select_db($database);

    $req="select max(timestamp) from valeurs_labo";
    $req = stripSlashes($req);
    $result = mysql_query($req);

    $tbl_ligne = mysql_fetch_row($result);
    $date = $tbl_ligne[0];

    $req="select V.nomlabo, octetin*100/(debit*1000), octetout*100/(debit*1000),
flotin*100/(NbFlotMax), flotout*100/(NbFlotMax), snmpin*100/(debit*1000),
snmpout*100/(debit*1000) from valeurs_labo V, laboratoires L where L.nomlabo=V.nomlabo and
timestamp = \"$date\"";
    $req = stripSlashes($req);
    $result = mysql_query($req);

    if ($result == 0)
    {
      echo("<B>Error toto " . mysql_errno() . ": " . mysql_error() . "</B>");
    }
    elseif (mysql_num_rows($result) == 0)
    {
      echo("<B>Operation effectuee!</B>");
    }
    else
```

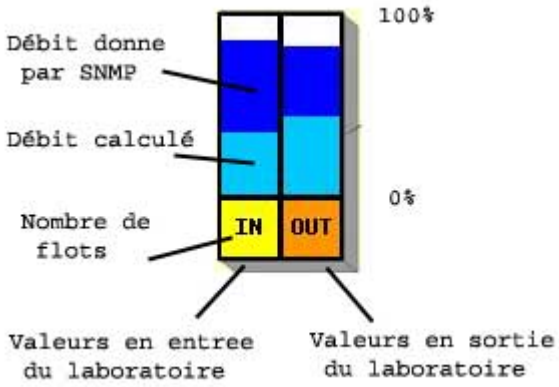
```
    {
      for ($i =0; $i < mysql_num_rows($result); $i++)
      {
        $tbl_ligne = mysql_fetch_row($result);
        $valeur["$tbl_ligne[0]"] = array($tbl_ligne[1], $tbl_ligne[2], $tbl_ligne[3],
$tbl_ligne[4], $tbl_ligne[5], $tbl_ligne[6]);
      }
    }
    mysql_close();
  ?>

  <?php
    function Gyro($nomlabo, $valeur, $date)
    {
      $alert_octin = 75;
      $alert_octout = 75;
      $alert_fluxin = 75;
      $alert_fluxout = 75;
      $alert_snmpin = 75;
      $alert_snmpout = 75;
      $elem = $valeur["$nomlabo"];
      if ($elem[0]<$alert_octin && $elem[1]<$alert_octout && $elem[2]<$alert_fluxin &&
$elem[3]<$alert_fluxout && $elem[4]<$alert_snmpin && $elem[5]<$alert_snmpout)
      {
        echo ("<IMG SRC=\"Images/Gyro_Vide.jpg\" WIDTH=63 HEIGHT=32></TD>\n");
      }
      else
      {
        echo ("<embed src=\"Images/Gyro.swf\" quality=high width=\"63\" height=\"32\"><!-
$nomlabo -></TD>\n");
        $user="netflow";
        $host="localhost";
        $password="netflow";
        $database="netflow";

        mysql_connect($host, $user, $password);
        mysql_select_db($database);

        $req="INSERT INTO alerte SELECT * FROM valeurs_labo WHERE nomlabo=\"$nomlabo\" AND
timestamp=\"$date\"";
        $req = stripSlashes($req);
        $result = mysql_query($req);

        mysql_close();

      }
    }
  ?>

  <?php
    function Labo($nomlabo, $valeur, $delta_tps, $popup)
    {
      $elem = $valeur["$nomlabo"];
      if ($popup == 1)
      {
        echo("<A HREF=\"./Plan.php?delta_tps=$delta_tps&popup=$popup\"
onClick=\"OuvPage('../Historique/Histo_labo.php?nomlabo=$nomlabo&delta_tps=$delta_tps','h_$nom
labo','scrollbars=1,width=940,height=600');\"
onMouseOver=\"OuvPage('Info.php?nomlabo=$nomlabo','info','toolbar=no,width=320,height=190');\"
onMouseOut=\"FermPage('Info.php?nomlabo=$nomlabo','info','toolbar=no,width=320,height=190');\"
>\n");
      }
      else
      {
        echo("<A HREF=\"./Plan.php?delta_tps=$delta_tps&popup=$popup\"
onClick=\"OuvPage('../Historique/Histo_labo.php?nomlabo=$nomlabo&delta_tps=$delta_tps','h_$nom
labo','scrollbars=1,width=940,height=600');\" >\n");
      }
      echo("<IMG
SRC=\"Images/Imagelabo.php?in_calcul=$elem[0]&out_calcul=$elem[1]&in_flux=$elem[2]&out_flux=$e
lem[3]&in_snmp=$elem[4]&out_snmp=$elem[5]\" WIDTH=63 HEIGHT=124 BORDER=0>");
      echo("</A></TD>");
    }
  ?>
```

… … …

```
    <TD COLSPAN=7 ROWSPAN=11 ALIGN = "CENTER" BACKGROUND="Images/Informations.jpg">
      <TABLE BORDER=2 BGCOLOR="#FFFFFF" WIDTH=200 HEIGHT=60>
      <TR>
        <TD ALIGN ="CENTER"><B> D&eacute;lais entre les valeurs : <BR>
        <?php echo $delta_tps ?> sec </B></TD>
      </TR>
      </TABLE>
      <BR>
      <TABLE BORDER=2 BGCOLOR="#FFFFFF" WIDTH=200 HEIGHT=40>
      <TR>
        <TD ALIGN = "CENTER"><B><?php echo $date ?></B></TD>
      </TR>
      </TABLE>
      <BR>
      <INPUT TYPE="Button" VALUE="Informations"
onClick="OuvPage('../Informations/Informations.php?delta_tps=<?php echo $delta_tps
?>','Informations','scrollbars=1,width=920,height=600')";>

<!--      <IMG SRC="Images/Informations.jpg" WIDTH=261 HEIGHT=215></TD> -->
    </TD>
… … …
    <TD>
      <?php Gyro("CRTBT", $valeur, $date); ?>
<!--       <IMG SRC="Images/Gyro_CRTBT.jpg" WIDTH=63 HEIGHT=32></TD> -->
    <TD ROWSPAN=3>
      <IMG SRC="Images/Plan_43.jpg" WIDTH=29 HEIGHT=186></TD>
    <TD COLSPAN=2>
      <?php Gyro("Services", $valeur, $date); ?>
<!--       <IMG SRC="Images/Gyro_Services.jpg" WIDTH=63 HEIGHT=32></TD> -->
    <TD COLSPAN=3 ROWSPAN=3>
      <IMG SRC="Images/Plan_45.jpg" WIDTH=75 HEIGHT=186></TD>
    <TD COLSPAN=3>
      <?php Gyro("LNeel", $valeur, $date); ?>
<!--       <IMG SRC="Images/Gyro_LNeel.jpg" WIDTH=63 HEIGHT=32></TD> -->
    <TD COLSPAN=2 ROWSPAN=3>
      <IMG SRC="Images/Plan_47.jpg" WIDTH=29 HEIGHT=186></TD>
    <TD COLSPAN=2>
      <?php Gyro("LEPES", $valeur, $date); ?>
<!--       <IMG SRC="Images/Gyro_LEPES.jpg" WIDTH=63 HEIGHT=32></TD> -->
    <TD COLSPAN=2 ROWSPAN=3>
      <IMG SRC="Images/Plan_49.jpg" WIDTH=28 HEIGHT=186></TD>
    <TD COLSPAN=2>
      <?php Gyro("CRISTALLO", $valeur, $date); ?>
<!--       <IMG SRC="Images/Gyro_Cristallo.jpg" WIDTH=63 HEIGHT=32></TD> -->
    <TD ROWSPAN=3>
      <IMG SRC="Images/Plan_51.jpg" WIDTH=29 HEIGHT=186></TD>
… … …
    <TD>
      <?php Labo("CRTBT", $valeur, $delta_tps, $popup); ?>
<!--                 <A HREF="../histo_labo.php?nomlabo=CRTBT">
        <IMG SRC="Images/CRTBT.jpg" WIDTH=63 HEIGHT=124 BORDER=0></A></TD> -->
    <TD COLSPAN=2>
      <?php Labo("Services", $valeur, $delta_tps, $popup); ?>
<!--                 <A HREF="../histo_labo.php?nomlabo=Services">
        <IMG SRC="Images/Services.jpg" WIDTH=63 HEIGHT=124 BORDER=0></A></TD> -->
    <TD COLSPAN=3>
      <?php Labo("LNeel", $valeur, $delta_tps, $popup); ?>
<!--                 <A HREF="../histo_labo.php?nomlabo=LNeel">
        <IMG SRC="Images/LNeel.jpg" WIDTH=63 HEIGHT=124 BORDER=0></A></TD> -->
    <TD COLSPAN=2>
      <?php Labo("LEPES", $valeur, $delta_tps, $popup); ?>
<!--                 <A HREF="../histo_labo.php?nomlabo=LEPES">
        <IMG SRC="Images/LEPES.jpg" WIDTH=63 HEIGHT=124 BORDER=0></A></TD> -->
    <TD COLSPAN=2>
      <?php Labo("CRISTALLO", $valeur, $delta_tps, $popup); ?>
<!--                 <A HREF="../histo_labo.php?nomlabo=Cristallo">
        <IMG SRC="Images/Cristallo.jpg" WIDTH=63 HEIGHT=124 BORDER=0></A></TD> -->
… … …
</BODY>
</HTML>
```

B. Image_labo.php

1. Copie d'écran

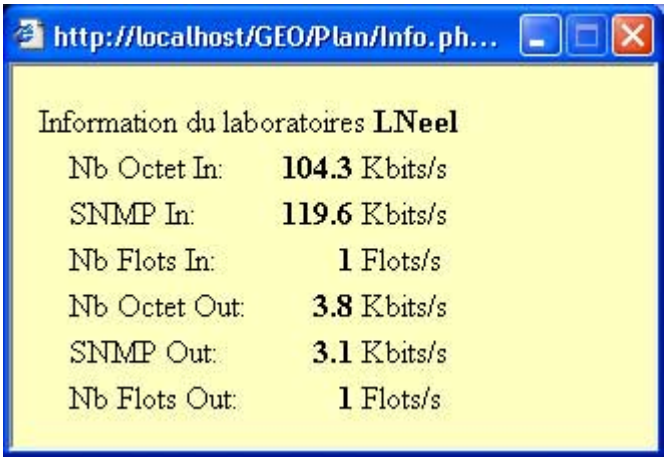


2. Code

```
<?php

  Header("Content-Type: image/jpeg");

  $largeur = 63;
  $hauteur = 124;

  $img = ImageCreate($largeur, $hauteur);

  $noir = ImageColorAllocate($img, 0, 0, 0);
  $blanc = ImageColorAllocate($img, 255, 255, 255);
  $vert = ImageColorAllocate($img, 110, 241, 118);
  $jaune = ImageColorAllocate($img, 255, 255, 0);
  $orange = ImageColorAllocate($img, 255, 150, 0);
  $rouge = ImageColorAllocate($img, 255, 0, 0);
  $bleuF = ImageColorAllocate($img, 0, 0, 255);
  $bleuC = ImageColorAllocate($img, 0, 200, 255);

  if ($largeur%2 == 0)
    $l_int = ($largeur-6)/2;
  else
    $l_int = ($largeur-7)/2;

  $h_int = $hauteur-7-$l_int;


  // Zone haut gauche
  if ($in_snmp > 100) $in_snmp=100;
  if ($in_calcul > 100) $in_calcul=100;
  ImageFilledRectangle ($img, 2, 2, $l_int+1, $h_int+1, $blanc);
  $h_rs = $h_int * $in_snmp / 100;
  ImageFilledRectangle ($img, 2, $h_int+1-$h_rs, $l_int+1, $h_int+1, $bleuF);
  $h_rc = $h_int * $in_calcul / 100;
  if ($h_rs = $h_rc) $h_rc = $h_rc -1;
  ImageFilledRectangle ($img, 2, $h_int+1-$h_rc, $l_int+1, $h_int+1, $bleuC);


  // Zone haut droite
  if ($out_snmp > 100) $out_snmp=100;
  if ($out_calcul > 100) $out_calcul=100;
  ImageFilledRectangle ($img, $largeur-$l_int-2, 2, $largeur-3, $h_int+1, $blanc);
  $h_rs = $h_int * $out_snmp / 100;
  ImageFilledRectangle ($img, $largeur-$l_int-2, $h_int+1-$h_rs, $largeur-3, $h_int+1,
$bleuF);
  $h_rc = $h_int * $out_calcul / 100;
  if ($h_rs = $h_rc) $h_rc = $h_rc -1;
  ImageFilledRectangle ($img, $largeur-$l_int-2, $h_int+1-$h_rc, $largeur-3, $h_int+1,
$bleuC);
```

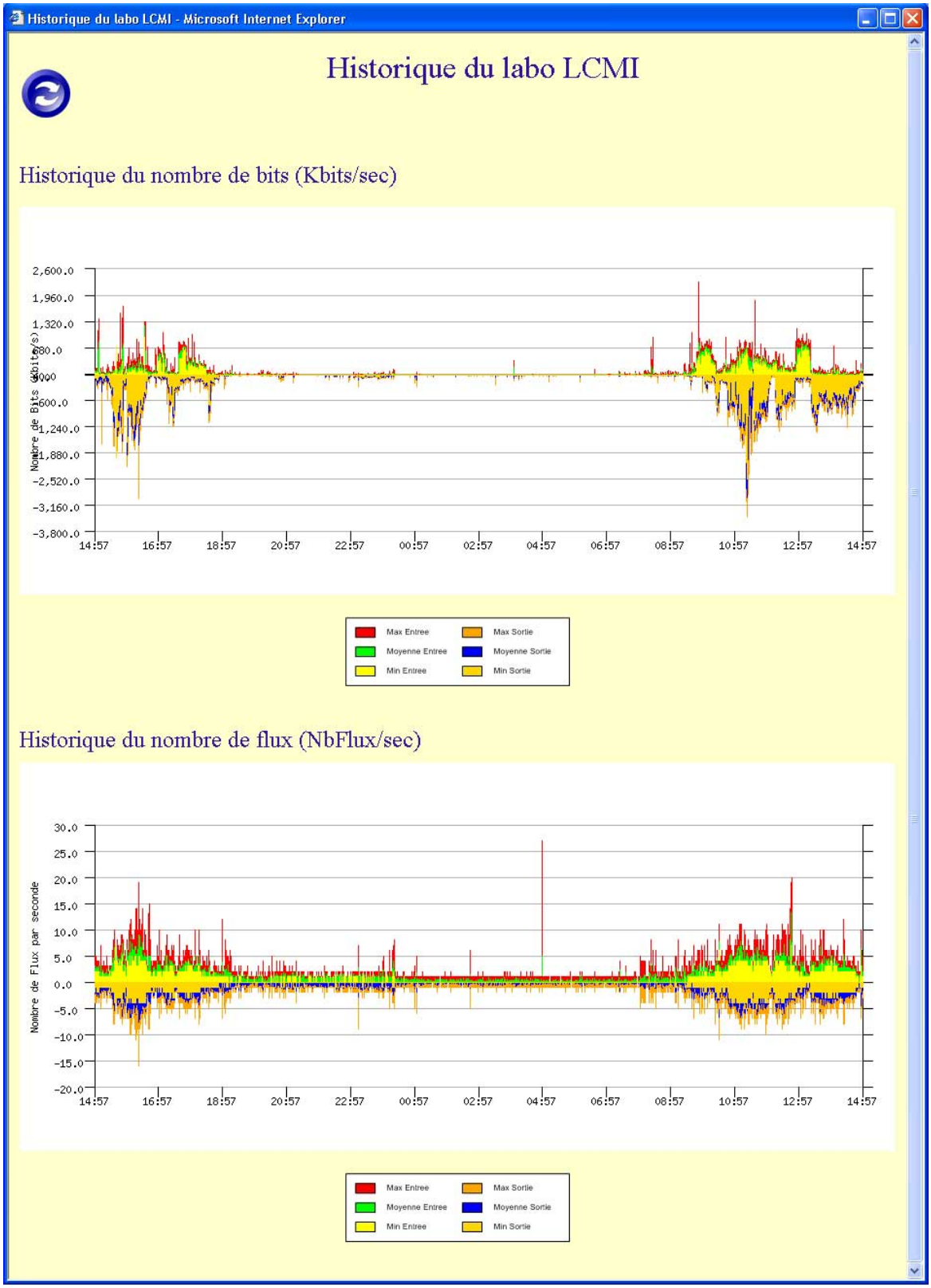
```
  // Zone bas gauche
  if ($in_flux == 0)
    $couleur = $blanc;
  elseif ($in_flux <= 25)
    $couleur = $vert;
  elseif ($in_flux <= 50)
    $couleur = $jaune;
  elseif ($in_flux <= 75)
    $couleur = $orange;
  else
    $couleur = $rouge;
  ImageFilledRectangle ($img, 2, $hauteur-2-$l_int, $l_int+1, $hauteur-3, $couleur);


  // Zone bas droite
  if ($out_flux == 0)
    $couleur = $blanc;
  elseif ($out_flux <= 25)
    $couleur = $vert;
  elseif ($out_flux <= 50)
    $couleur = $jaune;
  elseif ($out_flux <= 75)
    $couleur = $orange;
  else
    $couleur = $rouge;
  ImageFilledRectangle ($img, $largeur-$l_int-2, $hauteur-2-$l_int, $largeur-3, $hauteur-3,
$couleur);


  ImageString($img, 3, 9, $hauteur-24, "IN", $noir);
  ImageString($img, 3, $largeur-$l_int+2, $hauteur-24, "OUT", $noir);

  ImageJpeg($img);
  ImageDestroy($img);
?>
```

C. Info.php



III. Historique

A. Histo_labo.php



B. Histo_octet

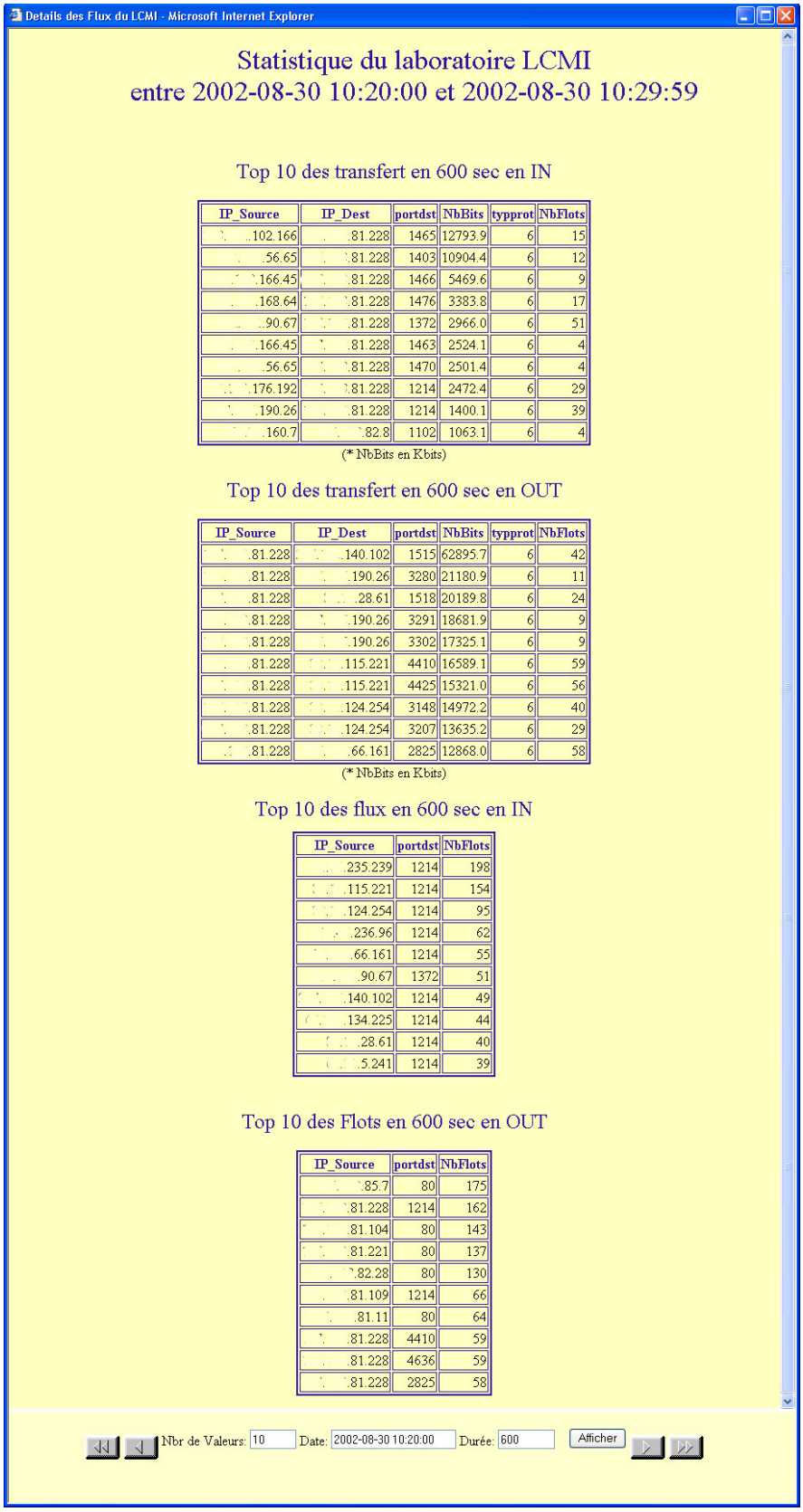
1. Code

```
<?php

  $user="netflow";
  $host="localhost";
  $password="netflow";
  $database="netflow";

  mysql_connect($host, $user, $password);
  mysql_select_db($database);
  $req="select FLOOR(UNIX_TIMESTAMP(timestamp)/60)*60 AS Date, AVG(octetin), MAX(octetin),
MIN(octetin), AVG(octetout), MAX(octetout), MIN(octetout) from valeurs_labo where
nomlabo=\"$nomlabo\" group by Date";

  $req = stripSlashes($req);
  $result = mysql_query($req);

  if ($result == 0):
    echo("<B>Error toto " . mysql_errno() . ": " . mysql_error() . "</B>");
  elseif (mysql_num_rows($result) == 0):
    echo("<B>Operation effectuee!</B>");
  else :
    for ($i =0; $i < mysql_num_rows($result); $i++)
    {
      $tbl_ligne = mysql_fetch_row($result);
      $new_avgoctetin=$tbl_ligne[1]/1000;
      $new_maxoctetin=$tbl_ligne[2]/1000;
      $new_minoctetin=$tbl_ligne[3]/1000;
      $new_avgoctetout=$tbl_ligne[4]*(-1)/1000;
      $new_maxoctetout=$tbl_ligne[5]*(-1)/1000;
      $new_minoctetout=$tbl_ligne[6]*(-1)/1000;
      $octet_data[$i] = array("",$tbl_ligne[0], $new_maxoctetin, $new_avgoctetin,
$new_minoctetin, $new_maxoctetout, $new_avgoctetout, $new_minoctetout);
    }

    $req="select CEILING(MAX(octetin)*1.1/100000)*100 from valeurs_labo where
nomlabo=\"$nomlabo\" group by nomlabo;";
    $req = stripSlashes($req);
    $result = mysql_query($req);
    $tbl_ligne = mysql_fetch_row($result);
    $maxoctet = $tbl_ligne[0];

    $req="select CEILING(MAX(octetout)*1.1/100000)*-100 from valeurs_labo where
nomlabo=\"$nomlabo\" group by nomlabo;";
    $req = stripSlashes($req);
    $result = mysql_query($req);
    $tbl_ligne = mysql_fetch_row($result);
    $minoctet = $tbl_ligne[0];

    //include("/usr/web/server/apache/phplot-4.4.6/phplot.php");
    include("../../phplot-4.4.6/phplot.php");

    $graph = new PHPlot(900,400);
    $graph->SetDataType("data-data");  //Must be called before SetDataValues
    $graph->SetXGridLabelType("time");
    $graph->SetXDataLabelAngle(45);
    $graph->SetBackgroundColor(array(255,255,255));
    $graph->SetXLabel("");
    $graph->SetYLabel("Nombre de Bits (Kbits/s)");
    $graph->SetXTimeFormat("%H:%M");
    $graph->SetDataValues($octet_data);
    $graph->SetHorizTickIncrement(7200);
    $graph->SetPlotType("thinbarline");
    $graph->SetPlotAreaWorld($octet_data[0][1],$minoctet,$octet_data[$i-1][1],$maxoctet);
    $graph->SetDataColors(array("red", "green", "yellow", "orange", "blue",
"gold"),array("black"));
    $graph->DrawGraph();

  endif
?>
```

C. Histo_flux

IV. Details

A. Details.php

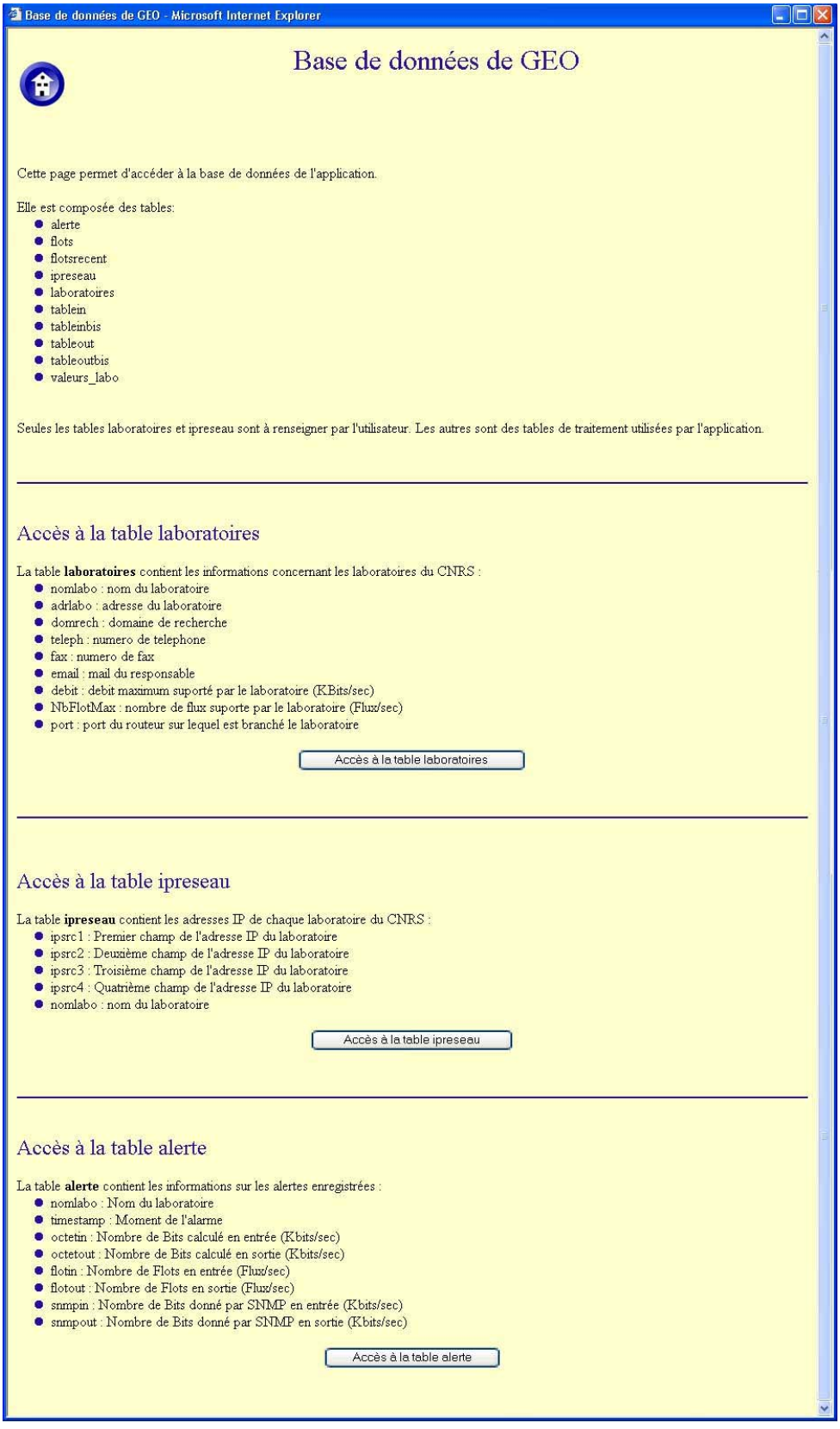


B. Niveau1.php

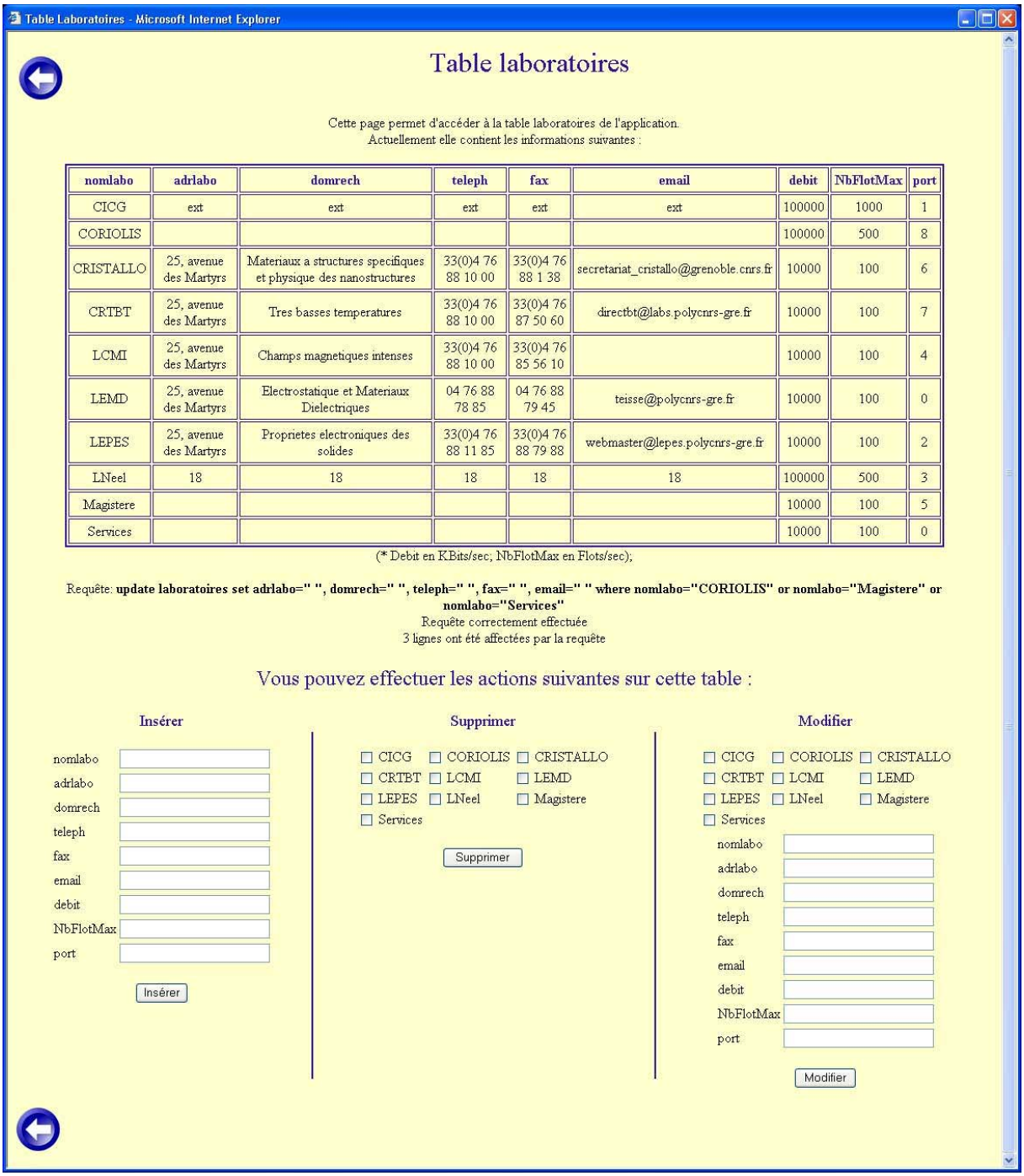
C. Navigation.php

V. Basededonnes

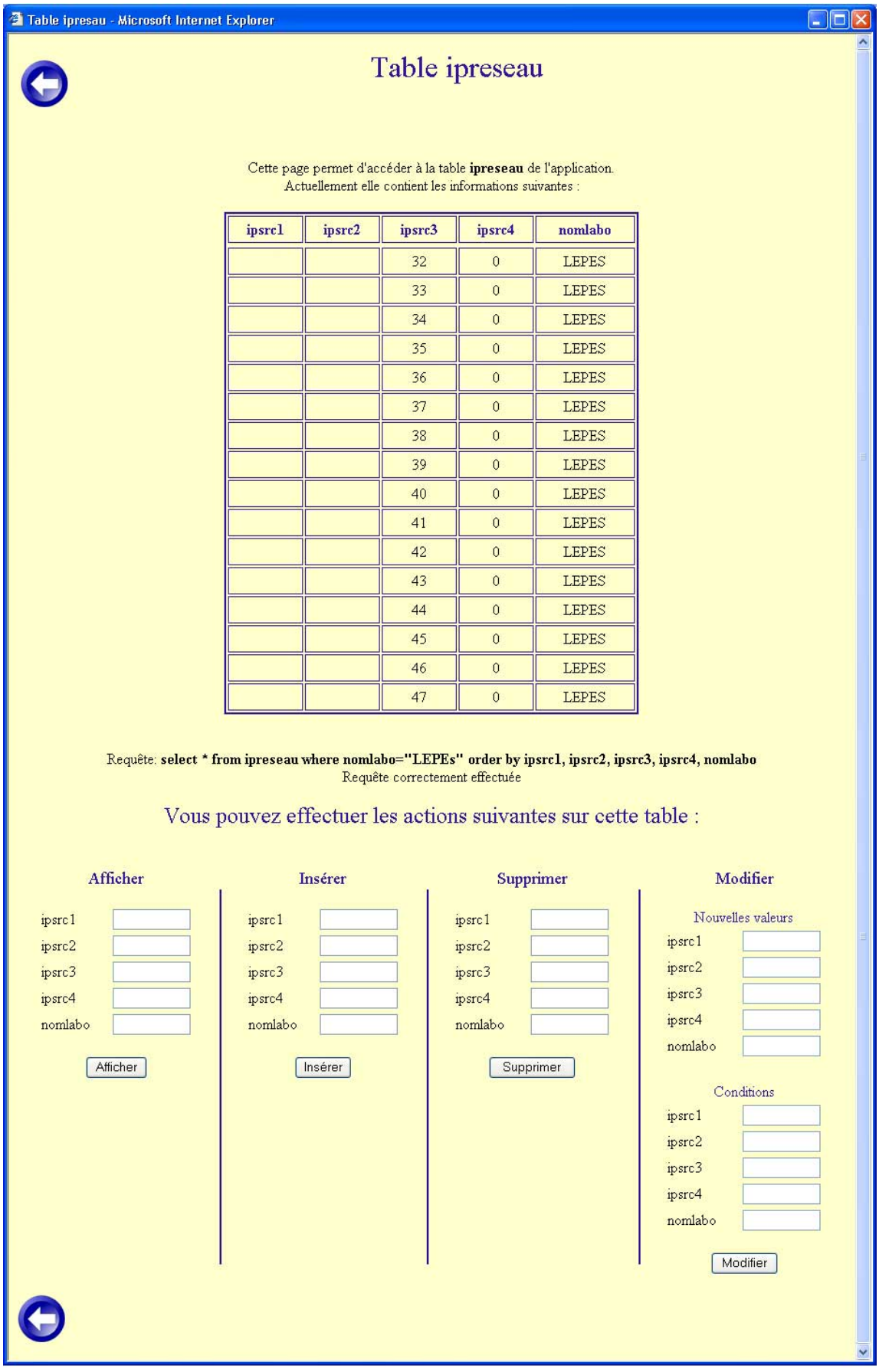
A. Basededonnes.php



B. Laboratoires.php



C. Ipreseau.php



D. Alerte.php

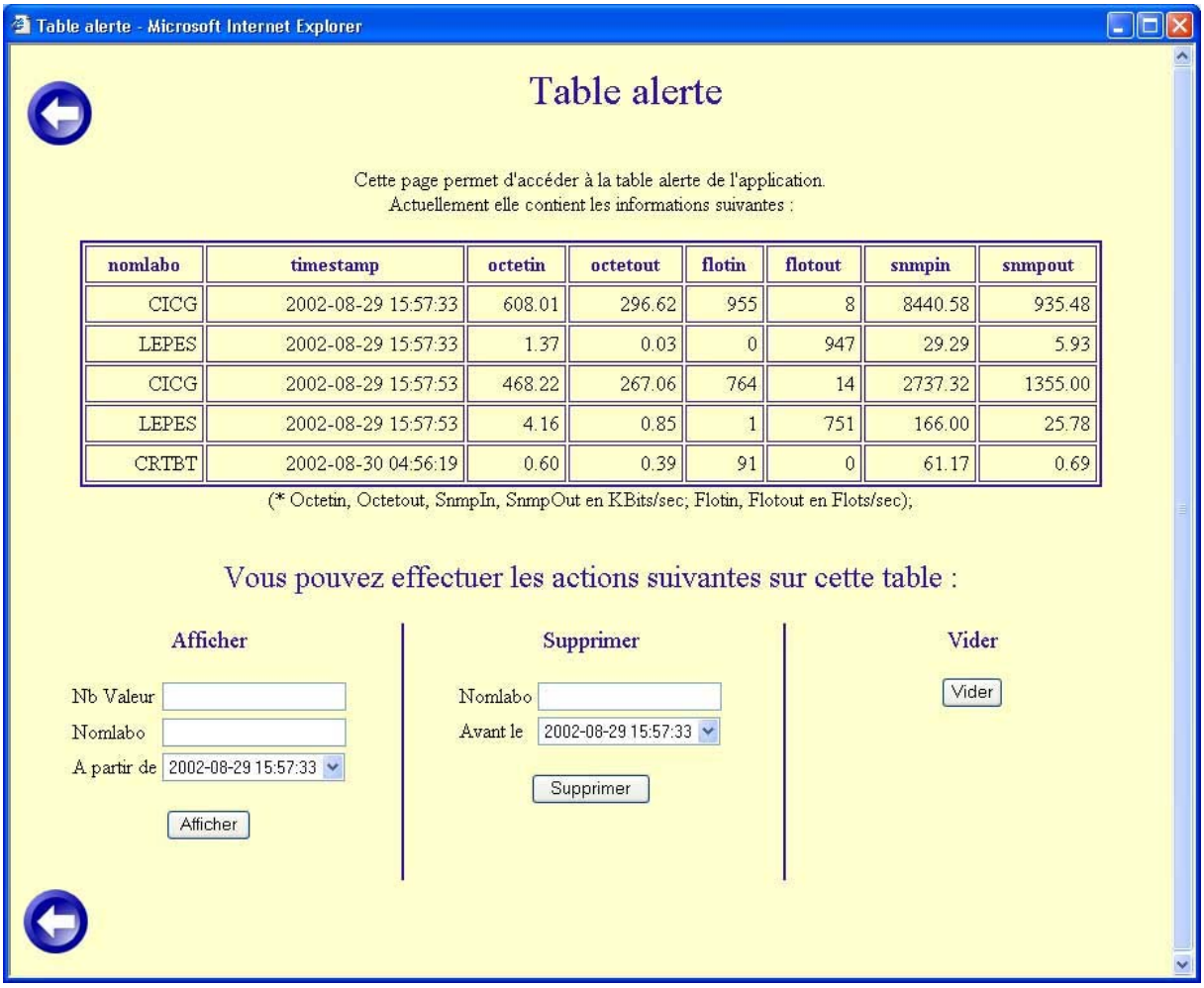


Table alerte - Microsoft Internet Explorer
Table alerte
Cette page permet d'accéder à la table alerte de l'application.
Actuellement elle contient les informations suivantes :
nomlabo | timestamp | octetin | octetout | flotin | flotout | snmpin | snmpout
CICG | 2002-08-29 15:57:33 | 608.01 | 296.62 | 955 | 8 | 8440.58 | 935.48
LEPES | 2002-08-29 15:57:33 | 1.37 | 0.03 | 0 | 947 | 29.29 | 5.93
CICG | 2002-08-29 15:57:53 | 468.22 | 267.06 | 764 | 14 | 2737.32 | 1355.00
LEPES | 2002-08-29 15:57:53 | 4.16 | 0.85 | 1 | 751 | 166.00 | 25.78
CRTBT | 2002-08-30 04:56:19 | 0.60 | 0.39 | 91 | 0 | 61.17 | 0.69
(* Octetin, Octetout, SnmpIn, SnmpOut en KBits/sec; Flotin, Flotout en Flots/sec);
Vous pouvez effectuer les actions suivantes sur cette table :
Afficher
Nb Valeur
Nomlabo
A partir de 2002-08-29 15:57:33
Afficher
Supprimer
Nomlabo
Avant le 2002-08-29 15:57:33
Supprimer
Vider
Vider