Université de Strasbourg

Contiki-ng

Julien Montavont – montavont@unistra.fr

Contiki-OS

- Système d'exploitation pour objets contraints
- Créé par Adam Dunkels (2002)
- Protothread
  - Multi-threading
  - Event-driven programming
- Contiki-ng
  - Dernière version
    - Plus de threads « classiques », fusion couches MAC et RDC, …
    - Support d'une plus grande variété de matériels (ARM-MCU)

Organisation

- arch
  - arch/cpu → Informations sur MCU
  - arch/dev → Informations puces et périphériques
  - arch/platform → Pilotes
- os → Fichiers systèmes et bibliothèques
- tools → Outils pour flasher / déboguer / simuler
- examples → Exemples de programmes
- tests → Programmes de tests

Protothread (1)

- Thread léger sans pile d'exécution pour systèmes contraints en mémoire
  - Threads classiques disposent chacun d'une pile d'exécution
  - Protothreads utilisent une pile d'exécution unique et ne demandent que 2 à 12 octets de statut
- Ajout de fonctions bloquantes dans un système à événements asynchrones
  - Mutex / sémaphores cachés => Programmation simplifiée

Protothread (2)

- Avantages
  - Écrits en C => pas de code machine spécifique
  - N'utilisent pas des fonctions sujettes aux erreurs (ex : longjmp(…) => saut vers un contexte de pile sauvegardé)
  - Empreinte mémoire limitée (2 octets)
  - Peuvent être utilisés avec ou sans système d'exploitation
  - Fournissent une attente bloquante sans un multi-threading complet et sans changements de pile

Exemple – hello-world.c

```
#include "contiki.h"
#include <stdio.h> /* For printf() */

/*---------------------------------------------*/
PROCESS(hello_world_process, "Hello world process");
AUTOSTART_PROCESSES(&hello_world_process);

/*---------------------------------------------*/
PROCESS_THREAD(hello_world_process, ev, data)
{
    PROCESS_BEGIN();
    printf("Hello, world\n");
    PROCESS_END();
}
```

Concurrence - Processus

PROCESS (name, process_name)	Création d'un processus
PROCESS_THREAD (name, ev, data)	Corps d'un processus
PROCESS_BEGIN ()	Déclaration du début d'un processus
PROCESS_END ()	Déclaration de la fin d'un processus
PROCESS_EXIT ()	Terminaison du processus
PROCESS_WAIT_EVENT()	Attente d'un événement
PROCESS_WAIT_EVENT_UNTIL (c)	Attente d'un événement avec condition
PROCESS_WAIT_UNTIL (c)	Attente d'une condition
PROCESS_YIELD()	Équivalent à PROCESS_WAIT_EVENT
PROCESS_YIELD_UNTIL (c)	Équivalent à PROCESS_WAIT_UNTIL
PROCESS_PAUSE()	Met en pause le processus courant
PROCESS_PT_SPAWN (pt, thread)	Crée un processus depuis le processus courant

Exemple

`<contiki-ng>/examples/hello-world/`

Gestion du temps

```
// temps courant en TICK
clock_time_t clock_time()

// temps courant en secondes
unsigned long clock_seconds();

// attente pdt un nbre de TICK
void clock_wait(int delay);

// une seconde mesurée en TICK
CLOCK_SECOND;
```

Temporisateurs - Timer

```
// démarrer un timer
void timer_set(struct timer *t, clock_time_t interval);

// réinit un timer avec précédente valeur depuis
// sa date d'expiration
void timer_reset(struct timer *t);

// réinit un timer avec valeur précédente depuis
// temps courant
void timer_restart(struct timer *t);


// vérifier si un timer a expiré (renvoie vrai ou faux)
int timer_expired(struct timer *t);

// obtenir le temps restant avant expiration
clock_time_t timer_remaining(struct timer *t);
```

Exemple

```
struct timer timer_timer;
timer_set (&timer_timer, 3 * CLOCK_SECOND);

// vérif. manuelle de l'expiration du temporisateur
if (timer_expired (&timer_timer)) {
    t = clock_time ();
    printf ("timer expired: %lu \n", t);
}
```

Temporisateurs - Stimer

```
// démarrer un Stimer
void stimer_set(struct stimer *t, unsigned long interval);

// réinit un Stimer avec valeur précédente depuis sa date
// d'expiration
void stimer_reset(struct stimer *t);

// réinit un Stimer avec valeur précédente depuis temps
// courant
void stimer_restart(struct stimer *t);


// vérifier si un temporisateur a expiré
int stimer_expired(struct stimer *t);

// obtenir le temps restant avant expiration
unsigned long stimer_remaining(struct stimer *t);
```

Temporisateur - Etimer

```
// démarrer un Etimer
void etimer_set(struct etimer *t, clock_time_t interval);

// réinit un Etimer avec valeur précédente depuis sa date
// d'expiration
void etimer_reset(struct etimer *t);

// réinit. un Etimer avec valeur précédente depuis temps
// courant
void etimer_restart(struct etimer *t);

// stopper un Etimer
void etimer_stop(struct etimer *t);

// vérifier si un Etimer a expiré
int etimer_expired(struct etimer *t);

// vérifier s'il reste des Etimer actifs (non expirés)
int etimer_pending();

// obtenir la date d'expiration du prochain Etimer
clock_time_t etimer_next_expiration_time();
```

Exemple

```
struct etimer etimer_timer;
etimer_set(&etimer_timer, 3 * CLOCK_SECOND);


while(1){
    PROCESS_WAIT_EVENT_UNTIL(etimer_expired(&etimer_timer));
    etimer_reset(&etimer_timer);
}
```

Temporisateur - Ctimer

```
// démarrer un Ctimer
void ctimer_set(struct ctimer *c,
                clock_time_t t,
                void(*f)(void *),
                void *ptr);

// réinit un Ctimer avec valeur précédente depuis sa date
// d'expiration
void ctimer_reset(struct ctimer *t);

// réinit. un Ctimer avec valeur précédente depuis
// temps courant
void ctimer_restart(struct ctimer *t);

// arrêter un Ctimer
void ctimer_stop(struct ctimer *t);

// vérifier si un Ctimer a expiré
int ctimer_expired(struct ctimer *t);
```

Exemple

```
void my_callback() {
    printf("ctimer callback called\n");
}


struct ctimer ctimer_timer;

ctimer_set(&ctimer_timer,
           CLOCK_SECOND,
           my_callback,
           NULL);
```

Temporisateurs - Rtimer

```
// heure courante en TICK
RTIMER_NOW();

// nombre de TICK par secondes
RTIMER_SECOND;

// création et ordonnancement d'une tâche temps réel
int rtimer_set(struct rtimer * task,
               rtimer_clock_t time,
               rtimer_clock_t duration, /* non utilisé */
               rtimer_callback_t func,
               void *ptr);
```

Exemple

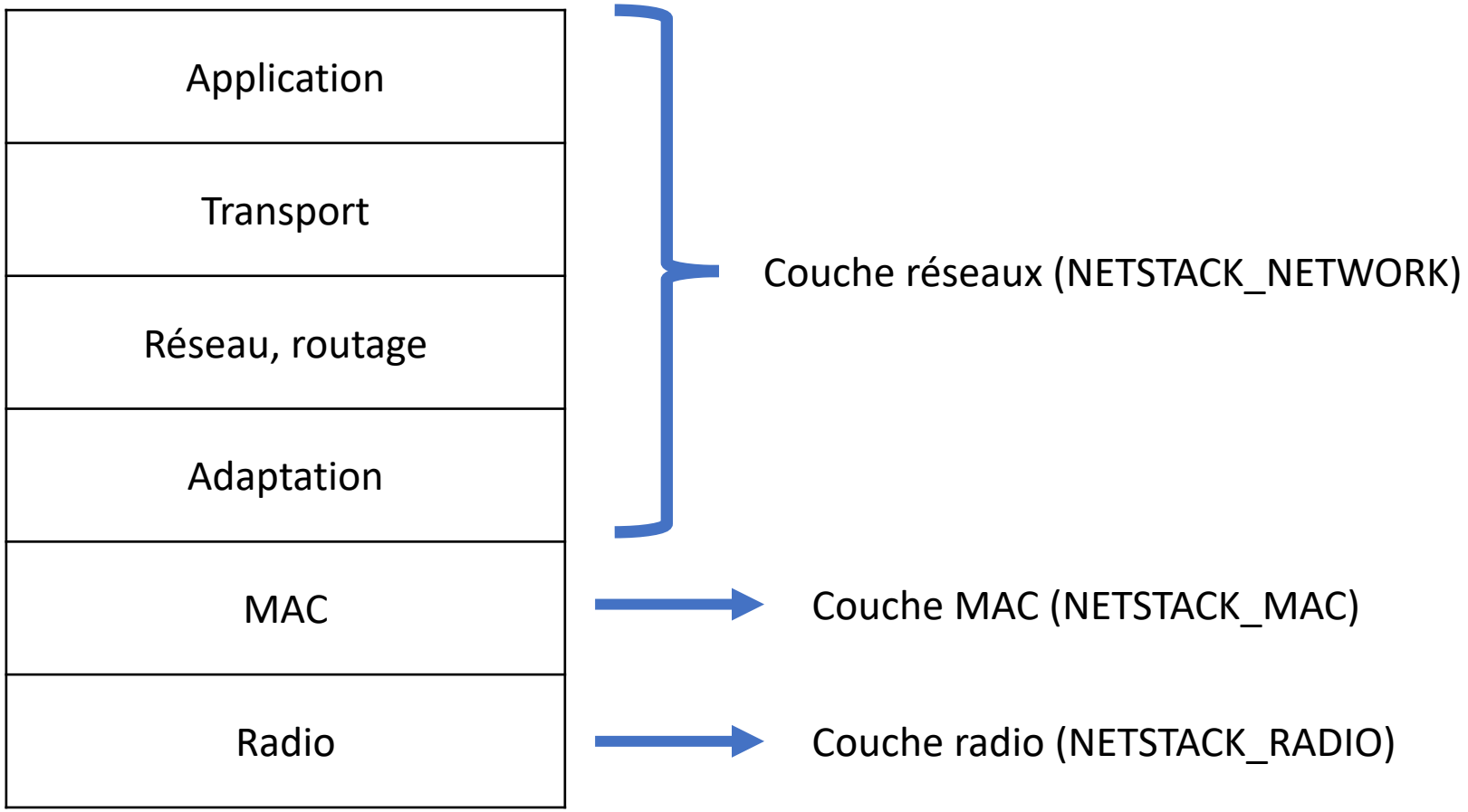
```
#include "sys/rtimer.h"

static rtimer_clock_t timeout_rtimer = RTIMER_SECOND / 2;

void callback(void) {
    printf("rtimer callback called\n");
}

struct rtimer task;
rtimer_set(&task,
           RTIMER_NOW() + timeout_rtimer,
           0,
           callback,
           NULL);
```

Pile réseau


Application
Transport
Réseau, routage
Adaptation
MAC
Radio
Couche réseaux (NETSTACK_NETWORK)
Couche MAC (NETSTACK_MAC)
Couche radio (NETSTACK_RADIO)

Bibliothèques réseaux

`<contiki-ng>/os/net`

- Couche applicative :
  - `http-socket.c, websocket.c, websocket-http-client.c, mqtt.c, coap.c, snmp.c`
- Couche transport :
  - `udp-socket.c` et `tcp-socket.c`
- Couche réseau et routage :
  - `uip6.c, rpl.c` et `nullrouting.c`
- Couche MAC :
  - `mac.c, csma.c, tsch.c` et `nullmac.c`

Communications UDP (1)

```
int simple_udp_send(struct simple_udp_connection *c,
                    const void *data,
                    uint16_t datalen)

int simple_udp_sendto(struct simple_udp_connection *c,
                      const void *data,
                      uint16_t datalen,
                      const uip_addr_t *to)

int simple_udp_sendto_port(struct simple_udp_connection *c,
                           const void *data,
                           uint16_t datalen,
                           const uip_ipaddr_t *to,
                           uint16_t to_port);
```

Communications UDP (2)

```
static void
udp_rx_callback (struct simple_udp_connection *c,
                 const uip_ipaddr_t *sender_addr,
                 uint16_t sender_port,
                 const uip_ipaddr_t *receiver_addr,
                 uint16_t receiver_port,
                 const uint8_t *data,
                 uint16_t datalen);


int
simple_udp_register(struct simple_udp_connection *c,
                    uint16_t local_port,
                    uip_ipaddr_t *remote_addr,
                    uint16_t remote_port,
                    simple_udp_callback receive_callback)
```

Exemple

`<contiki-ng>/examples/rpl-udp/`

Communication sans IP / Nullnet

- Paquets réseaux directement créés par l'application et transmis à la couche MAC
- À la réception, la couche MAC passe directement les paquets reçus à la couche Nullnet qui déclenche un callback applicatif

Exemple : émission

```
#include "net/nullnet/nullnet.h »

...
uint8_t payload [64] = { 0 };

/* Point NullNet buffer to payload */
nullnet_buf = payload;

/* Tell NullNet the payload length */
nullnet_len = 2;

/* Send in broadcast */
NETSTACK_NETWORK.output(NULL);

static linkaddr_t dest_addr = {{ 0x01, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00 }};
/* Send in unicast */
NETSTACK_NETWORK.output(&dest_addr);
```

Exemple : réception

```
#include "net/nullnet/nullnet.h »

...
void input_callback(const void *data,
                    uint16_t len,
                    const linkaddr_t *src,
                    const linkaddr_t *dest) {
    ...
} ... /* At process initialization */

nullnet_set_input_callback(input_callback);
```

Couche réseau

- Contiki-ng implémente IPv6
  - `uip.h` et `uip6.c` dans `<contiki-ng>/os/net/ipv6`
- 2 sous couches
  - IPv6 / adaptation layer (6LoWPAN / 6TOP)
- Routage assuré par
  - RPL
    - Construction d'un Destination Oriented Directed Acyclic Graph (DoDAG)
    - 2 version (full et light)
  - Nullrouting : pas de routage
- Nullnet : couche réseau qui ne fait rien
  - scénarios sans IPv6 ou tests des couches basses

Pilote générique de routage

<contiki-ng>/os/net/routing/routing.h

```
struct routing_driver {
  char *name;
  void (* init)(void);
  void (* root_set_prefix)(uip_ipaddr_t *prefix, uip_ipaddr_t *iid);
  int (* root_start)(void);
  int (* node_is_root)(void);
  int (* get_root_ipaddr)(uip_ipaddr_t *ipaddr);
  int (* get_sr_node_ipaddr)(uip_ipaddr_t *addr, const uip_sr_node_t
*node);
  int (* node_has_joined)(void);
  int (* node_is_reachable)(void);
  void (* global_repair)(const char *str);
  void (* local_repair)(const char *str);
  bool (* ext_header_remove)(void);
  int (* ext_header_update)(void);

   ...
}
```

Implémentation d'un pilote de routage

<contiki-ng>/os/net/routing/nullrouting/nullrouting.c

```
const struct routing_driver nullrouting_driver = {
  "nullrouting",
  init,
  root_set_prefix,
  root_start,
  node_is_root,
  get_root_ipaddr,
  get_sr_node_ipaddr,
  leave_network,
  node_has_joined,
  node_is_reachable,
  global_repair,
  local_repair,
  ext_header_remove,
  ext_header_update,
  ...
};
```

Couche MAC

`<contiki-ng>/os/net/mac`

- CSMA/CA en mode non beacon sur 802.15.4
- Time Slotted Channel Hopping (TSCH) sur 802.15.4
- Nullmac

Configuration pile réseau

- Dans le Makefile
  - Couche MAC

```
MAKE_MAC = MAKE_MAC_CSMA
MAKE_MAC = MAKE_MAC_TSCH
MAKE_MAC = MAKE_MAC_NULLMAC
```

  - Protocole de routage

```
MAKE_ROUTING = MAKE_ROUTING_RPL_LITE
MAKE_ROUTING = MAKE_ROUTING_RPL_CLASSIC
MAKE_ROUTING = MAKE_ROUTING_NULLROUTING
```

  - Couche Réseau

```
MAKE_NET = MAKE_NET_NULLNET
MAKE_NET = MAKE_NET_IPV6
```

Configuration du système

- Logs / adresses / constantes / etc.
- Création d'un fichier project-conf.h
  - make viewconf
  - <contiki-ng>/os/contiki-default-conf.h
  - Définition de constantes dans project-conf.h

```
#define LOG_CONF_LEVEL_MAC LOG_LEVEL_DBG
#define IEEE802154_CONF_PANID 0x8000
#define UIP_CONF_BUFFER_SIZE 160
#define ENERGEST_CONF_ON 1
…
```

Modules

- Fonctionnalités additionnelles
  - Protocoles (e.g. CoAP)
  - Ordonnanceur TSCH (e.g. Orchestra)
  - Shell interactif
  - …
- Exemple (dans le Makefile)

```
MODULES += $(CONTIKI_NG_SERVICES_DIR)/coap
MODULES += $(CONTIKI_NG_SERVICES_DIR)/orchestra
MODULES += $(CONTIKI_NG_SERVICES_DIR)/shell
```

Documentation supplémentaire

- API :

https://contiki-ng.readthedocs.io/en/master/_api/modules.html

- Documentation, tutoriels

https://github.com/contiki-ng/contiki-ng/wiki

- Contiki-ng cheat sheet

https://www.contiki-ng.org/resources/contiki-ng-cheat-sheet.pdf