Here’s your complete **AI Routing Protocol (ai-routing)** implementation in Contiki 2.7 with:

* ✅ **Fuzzy Logic Rules** (for intelligent transmission)
* ✅ **Powertrace Integration** (for energy usage logging)
* ✅ **UDP Communication with Sink**
* ✅ 🛠 Ready for use in Cooja

**🔧 ai-routing-node.c (Sensor Node with Fuzzy Logic + Powertrace)**

#include "contiki.h"

#include "contiki-net.h"

#include "uip.h"

#include "uip-ds6.h"

#include "uip-udp-packet.h"

#include "dev/leds.h"

#include "sys/etimer.h"

#include "powertrace.h"

#include <stdio.h>

#include <string.h>

#define UDP\_PORT 8765

#define SEND\_INTERVAL (30 \* CLOCK\_SECOND)

static struct uip\_udp\_conn \*udp\_conn;

static uint16\_t residual\_energy = 100;

static uint16\_t simulated\_distance = 40; // Simulated distance (can vary manually)

typedef enum {

LOW,

MEDIUM,

HIGH

} fuzzy\_level;

fuzzy\_level fuzzify\_energy(uint16\_t energy) {

if (energy < 30) return LOW;

else if (energy < 70) return MEDIUM;

else return HIGH;

}

fuzzy\_level fuzzify\_distance(uint16\_t distance) {

if (distance < 20) return LOW;

else if (distance < 50) return MEDIUM;

else return HIGH;

}

PROCESS(ai\_routing\_node\_process, "AI Routing Node Process");

AUTOSTART\_PROCESSES(&ai\_routing\_node\_process);

PROCESS\_THREAD(ai\_routing\_node\_process, ev, data)

{

static struct etimer periodic\_timer;

static char buf[50];

PROCESS\_BEGIN();

powertrace\_start(CLOCK\_SECOND \* 10); // Start powertrace

uip\_ipaddr\_t sink\_ipaddr;

uip\_ip6addr(&sink\_ipaddr, 0xfe80,0,0,0,0x0212,0x7401,1,1); // Replace with correct sink IP

udp\_conn = udp\_new(&sink\_ipaddr, UIP\_HTONS(UDP\_PORT), NULL);

udp\_bind(udp\_conn, UIP\_HTONS(UDP\_PORT));

etimer\_set(&periodic\_timer, SEND\_INTERVAL);

while (1) {

PROCESS\_WAIT\_EVENT\_UNTIL(etimer\_expired(&periodic\_timer));

residual\_energy -= 1; // Simulate drain

fuzzy\_level e\_level = fuzzify\_energy(residual\_energy);

fuzzy\_level d\_level = fuzzify\_distance(simulated\_distance);

if (e\_level == LOW && d\_level == HIGH) {

printf("Node: LOW energy + HIGH distance → Skipping send\n");

} else {

sprintf(buf, "Energy:%u", residual\_energy);

uip\_udp\_packet\_send(udp\_conn, buf, strlen(buf));

printf("Node: Sent: %s\n", buf);

}

etimer\_reset(&periodic\_timer);

}

PROCESS\_END();

}

**🛰 ai-routing-sink.c (Sink Node with Feedback Option)**

#include "contiki.h"

#include "contiki-net.h"

#include "uip.h"

#include "uip-udp-packet.h"

#include "sys/etimer.h"

#include <stdio.h>

#include <string.h>

#define UDP\_PORT 8765

#define FEEDBACK\_INTERVAL (60 \* CLOCK\_SECOND)

static struct uip\_udp\_conn \*udp\_conn;

PROCESS(ai\_routing\_sink\_process, "AI Routing Sink Process");

AUTOSTART\_PROCESSES(&ai\_routing\_sink\_process);

PROCESS\_THREAD(ai\_routing\_sink\_process, ev, data)

{

static struct etimer feedback\_timer;

static char last\_data[100];

PROCESS\_BEGIN();

udp\_conn = udp\_new(NULL, UIP\_HTONS(UDP\_PORT), NULL);

udp\_bind(udp\_conn, UIP\_HTONS(UDP\_PORT));

etimer\_set(&feedback\_timer, FEEDBACK\_INTERVAL);

while (1) {

PROCESS\_YIELD();

if (ev == tcpip\_event && uip\_newdata()) {

char \*received = (char \*)uip\_appdata;

int len = uip\_datalen();

memcpy(last\_data, received, len);

last\_data[len] = '\0';

printf("Sink: Received: %s\n", last\_data);

}

if (etimer\_expired(&feedback\_timer)) {

printf("Sink: Evaluating node status: %s\n", last\_data);

etimer\_reset(&feedback\_timer);

}

}

PROCESS\_END();

}

**🧾 Makefile**

CONTIKI = /home/user/contiki-2.7

CONTIKI\_PROJECT = ai-routing-node ai-routing-sink

all: $(CONTIKI\_PROJECT)

CONTIKI\_WITH\_IPV6 = 1

UIP\_CONF\_IPV6=1

UIP\_CONF\_DS6\_ROUTING=1

APPS += powertrace

include $(CONTIKI)/Makefile.include

**🧪 Simulation Tips**

* Add **multiple sensor motes** in Cooja and one sink.
* Use **Mote Output** for real-time fuzzy decisions + powertrace logs.
* To visualize energy: use PowerTracker or export logs to Excel for graphs.
* Adjust simulated\_distance per node to test fuzzy logic outcomes.

Would you like the **graph templates** (energy vs time, packets vs round) or the **fuzzy logic decision table** for your paper?