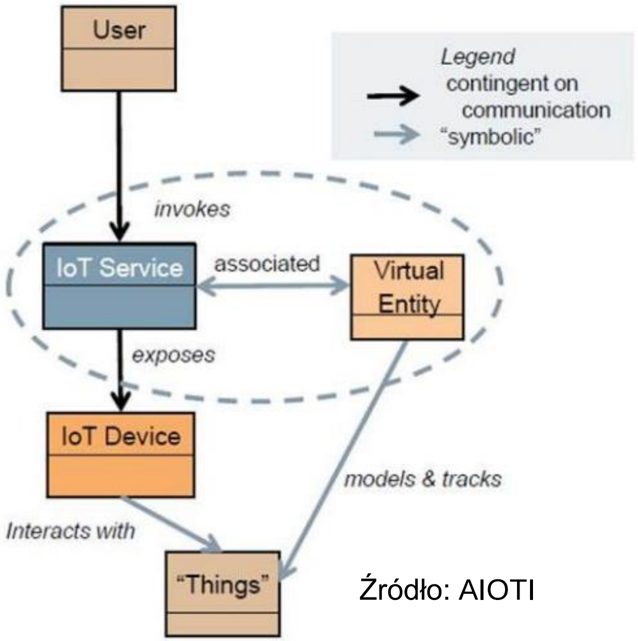
PSIR

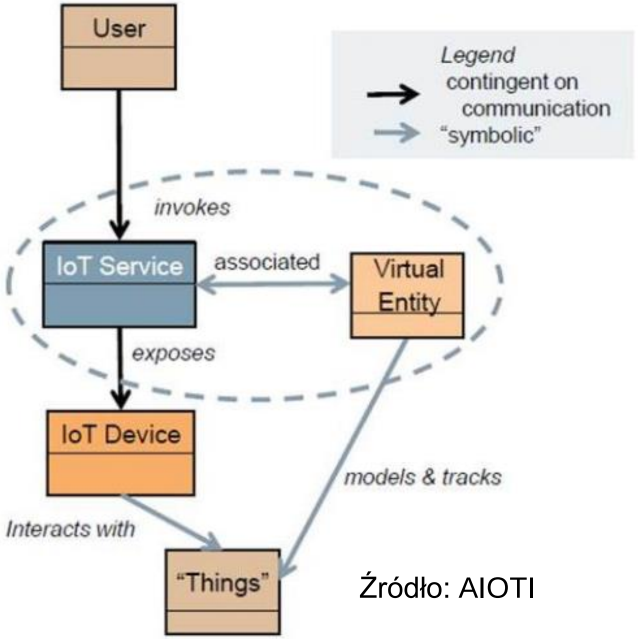
Wprowadzenie do projektu 2024Z: gra Penney's ante

Jarosław Domaszewicz
Instytut Telekomunikacji Politechniki Warszawskiej



Źródło: AIOTI

PSIR project concept and history



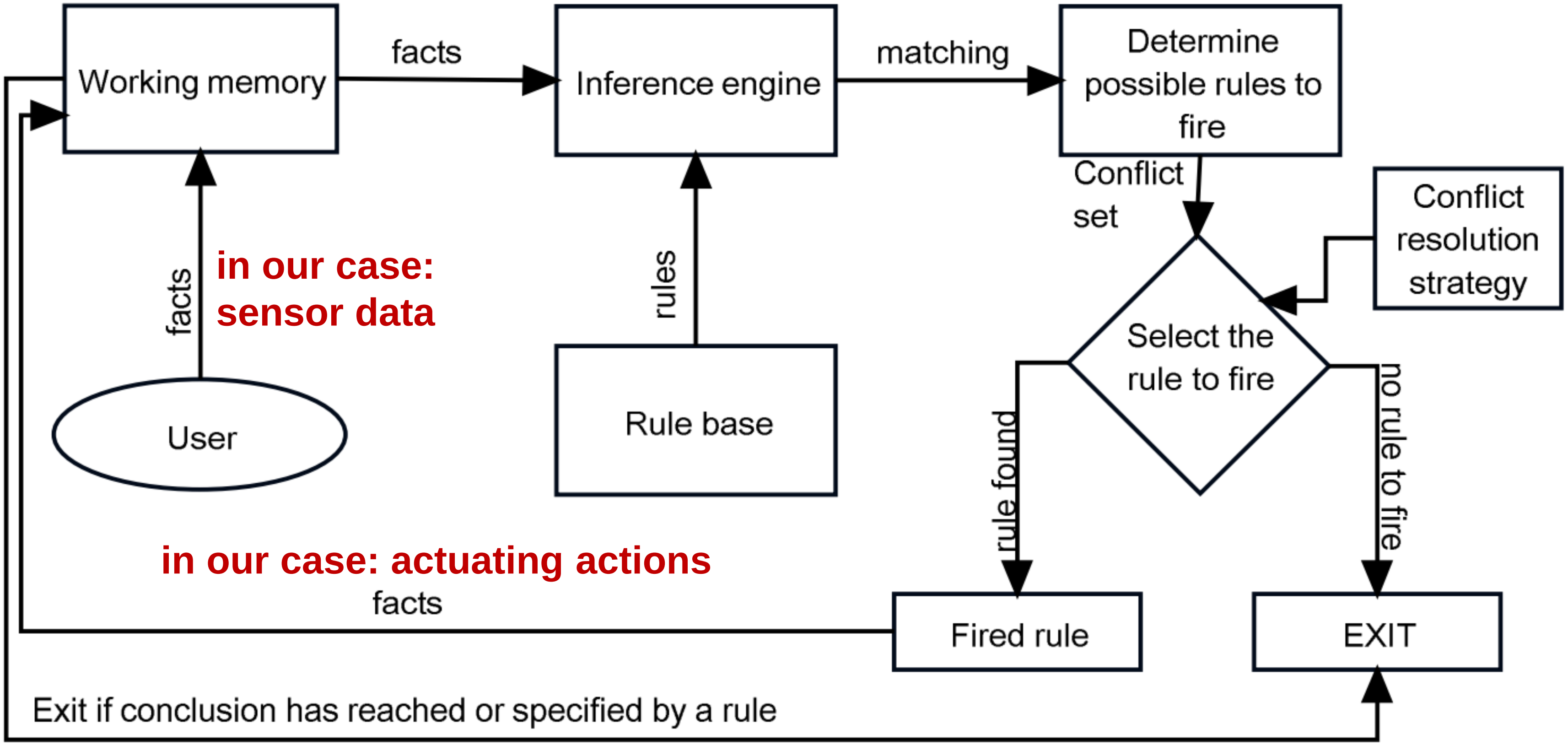
Źródło: AIOTI

PSIR PROJECT CONCEPT

A programming challenge that
helps develop solid programming skills
while being intellectually stimulating.

Enjoy stretching your mind!

21Z: A SMART ENVIRONMENT RULE-BASED SYSTEM

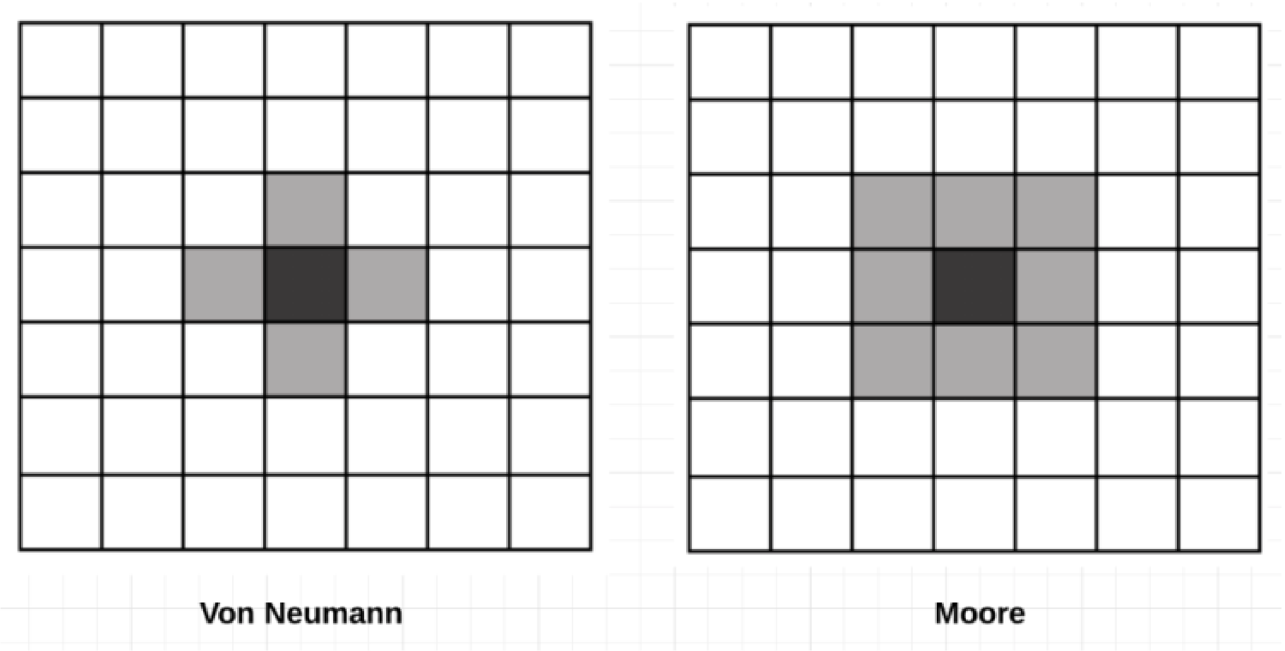


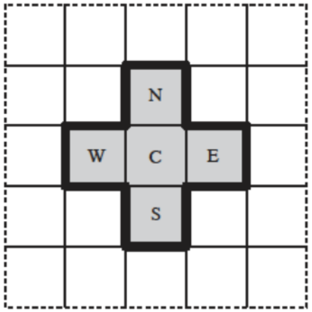
in our case: sensor data

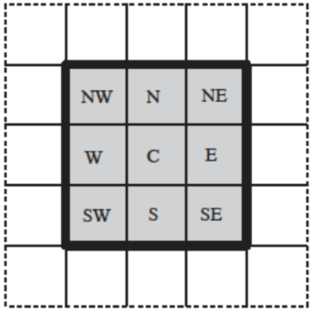
in our case: actuating actions

Source:
Intelligent Systems A Modern Approach,
Crina Grosan, Ajith Abraham,
Series: Intelligent Systems Reference Library, Springer,
DOI 10.1007/978-3-642-21004-4
Chapter 7, Rule-based Expert Systems, pp. 149-185

22L: Cellular automata with space partitioning





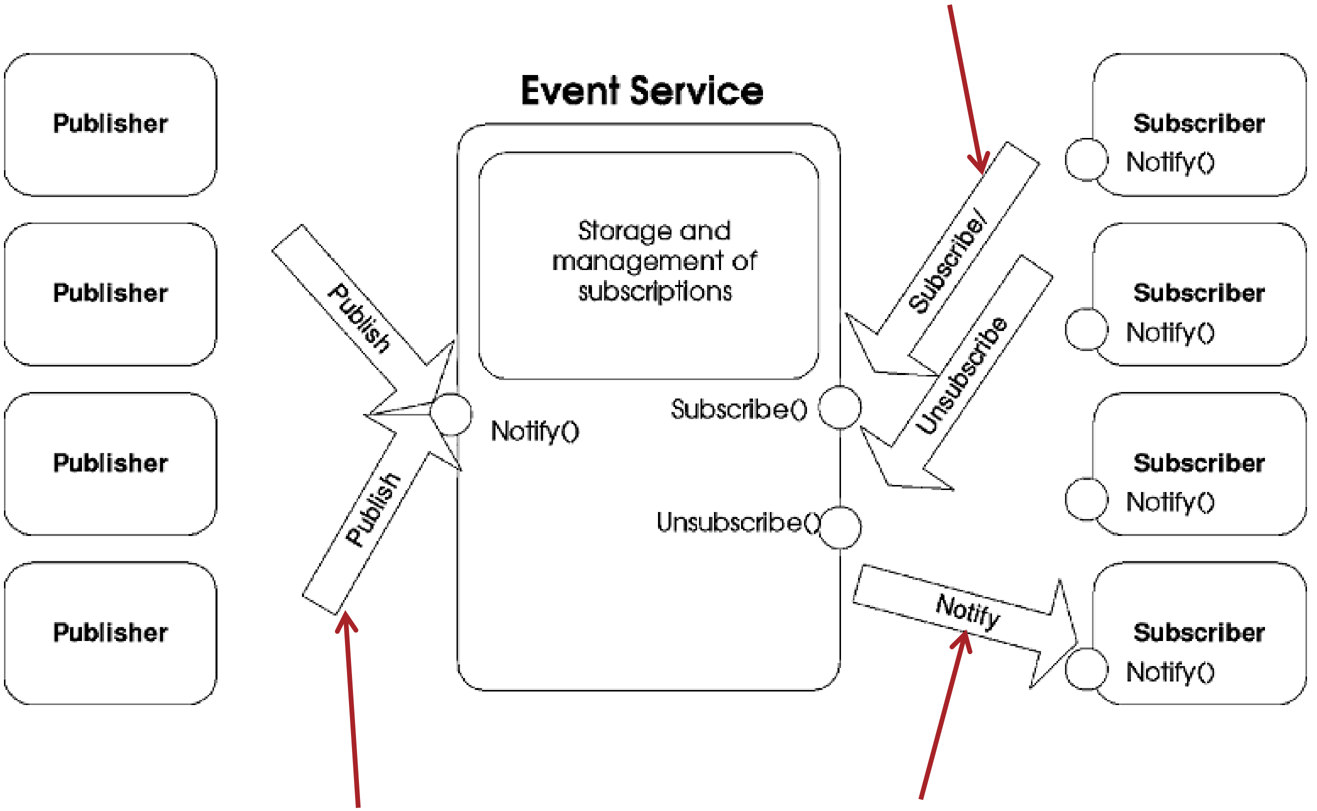


Source: Joseph Quartieri, Nikos E. Mastorakis, Gerardo Iannone, Claudio Guarnaccia
A Cellular Automata Model for Fire Spreading Prediction

Source: Washington Velasquez, Andres Munoz-Arcentales*y*, Thomas Michael Bohnertz, Joaquin Salvachua
Wildfire Propagation Simulation Tool using Cellular Automata and GIS

22Z: Content-based publish/subscribe

1. subscribers express interest in selected events



highly recommended

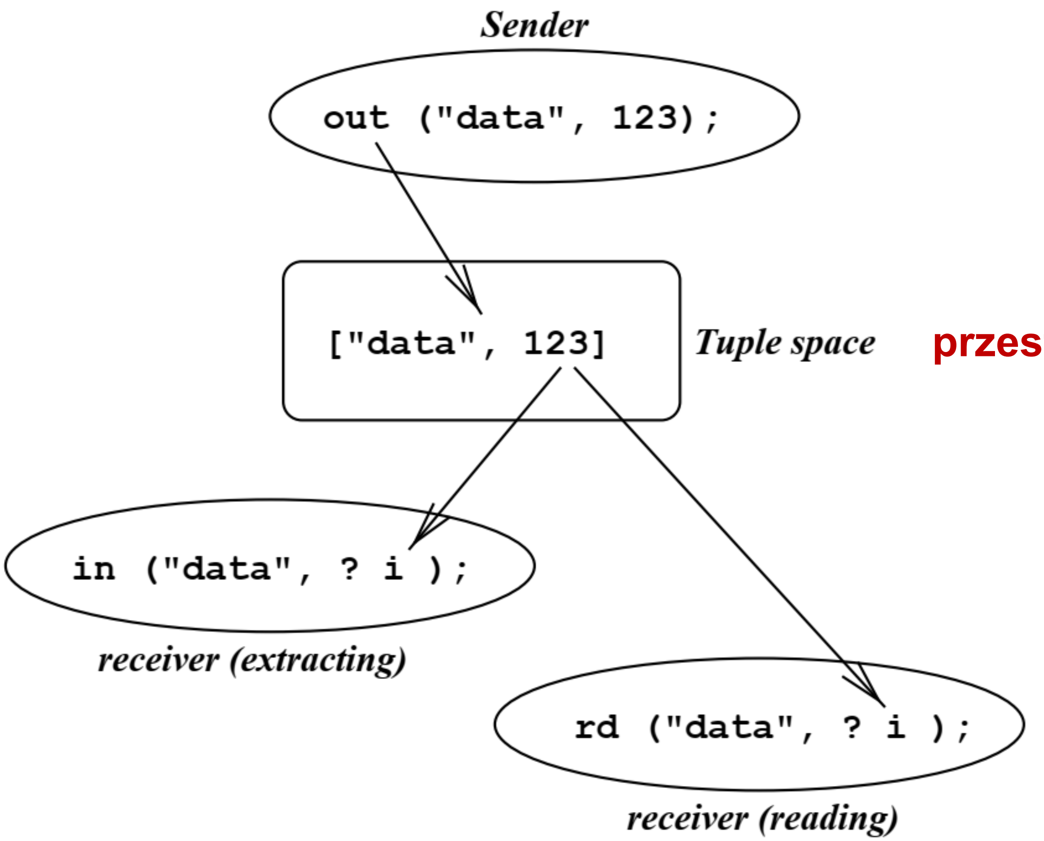
Source:
The Many Faces of Publish/Subscribe
P. Eugster et al.,
ACM Computing Surveys,
Vol. 35, No. 2, 2003

2. publishers produce assorted events

3. subscribers receive events they are interested in

- *publishers, subscribers, event service* (broker, server, middleware)
- note: push vs. pull, one-to-many, many-to-one

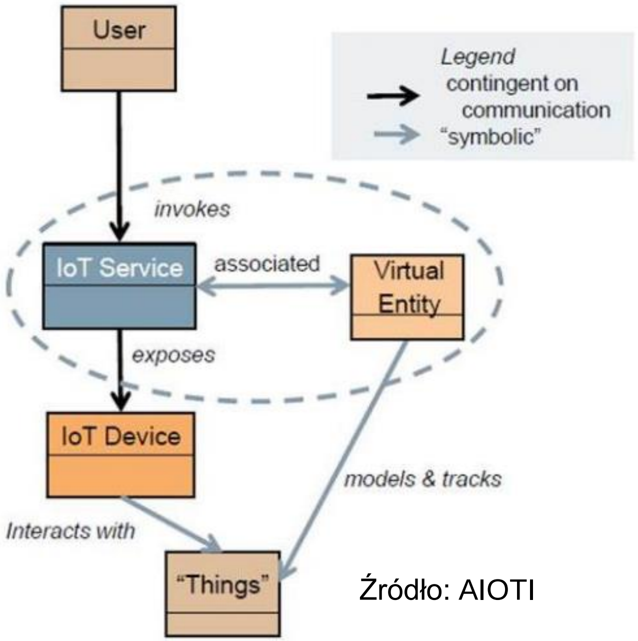
23Z: Tuple space middleware



przestrzeń krotek

W. Hasselbring and M. Roantree. 1998.
A Generative Communication Service for Database Interoperability.
In Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems (COOPIS '98). IEEE Computer Society, USA, 64–73.

Penney’s ante gane



Źródło: AIOTI

Source: https://en.wikipedia.org/wiki/Coin_flipping
Classical Numismatic Group, Inc. http://www.cngcoins.com

Navia aut caput

Coin flipping, **coin tossing**, or **heads or tails**
is the practice of throwing a coin in the air
and checking which side is showing when it lands,
in order to randomly choose between two alternatives.

Coin flipping was known to the Romans as *navia aut caput* ("ship or head").

Source: Isha Agarwal, Matvey Borodin, Aidan Duncan, Kaylee Ji, Tanya Khovanova, Shane Lee, Boyan Litchev, Anshul Rastogi, Garima Rastogi, Andrew Zhao
From Unequal Chance to a Coin Game Dance: Variants of Penney's Game, https://doi.org/10.48550/arXiv.2006.13002, edited.

PENNEY’S ANTE GAME

Alice and Bob have a fair coin to flip.
They have decided to play a game known as **Penney’s game.**

Alice selects a pattern of heads (H) and tails (T) of length *n*,
after which Bob chooses his own pattern of heads and tails, also of length *n*.
They then begin tossing the coin.
Whoever’s pattern appears first in the sequence of heads and tails is the winner.

Suppose *n* = 3, and Alice selects HHH, while Bob selects THH.
They toss the coin several times and get H H T T H T H H.
We see that Bob is the winner of this round.

Some interesting questions

- Recall: Alice selects HHH, while Bob selects THH

- Are Bob and Alice equally likely to win?

- What is the probability that Alice wins?
- What is the probability that Bob wins?

- What is, on average, the number of coin tosses needed to complete a game?

- What about different combinations of patterns and different lengths of patterns (*n*)?
- What if there are more than two players?

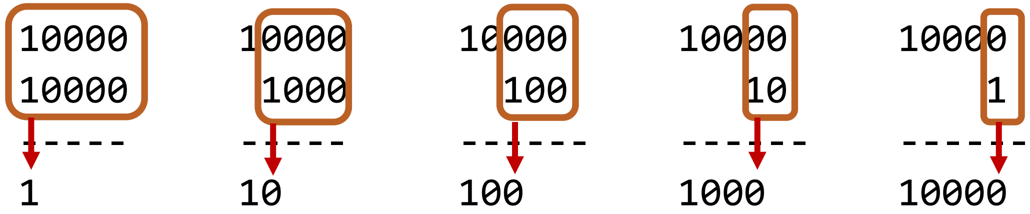
Source: Stanley Collings
Coin sequence probabilities and paradoxes. Bulletin of the Institute of Mathematics and its Applications (1982) 18, 227-232.

A SIMPLE EXAMPLE

- Alice picks HH, and Bob picks TH.
- The probability of each pattern is ¼, so the game appears fair.
- Is it?
- Consider the results of the first two tosses.
  - HH… -> Alice wins
  - TH… -> Bob wins
  - HT… -> can Alice win?
  - TT… -> can Alice win?
- Probability of Alice winning is ¼.
- Probability of Bob winning is ¾.

MAGIC OF CONWAY LEADING NUMBERS (1/3)

- Let w_1 and w_2 be two patterns.
- Define a Conway leading number as in the following example.
- $w_1 = 10000, w_2 = 11111$
- C^{w_1,w_1}



```
10000      10000      10000      10000      10000
10000       1000        100         10          1
-----     ------     -------    -------    -------
1          10         100        1000       10000
```

- Keep shifting the second argument with respect to the first argument and checking if the prefix of the second argument is the same as the suffix of the first argument.
- Let's calculate C^{w_1,w_2} , C^{w_2,w_1} , and C^{w_2,w_2}

Magic of Conway leading numbers (2/3)

- Let P_1 and P_2 be the probabilities that w_1 and w_2 occurs first, respectively. Then

$$\frac{P_1}{P_2} = \frac{C^{2,2} - C^{2,1}}{C^{1,1} - C^{1,2}}$$

- In our example ($w_1 = 10000, w_2 = 11111$):

$$\frac{P_1}{P_2} = \frac{31 - 1}{16 - 0} = \frac{30}{16} = \frac{15}{8}$$

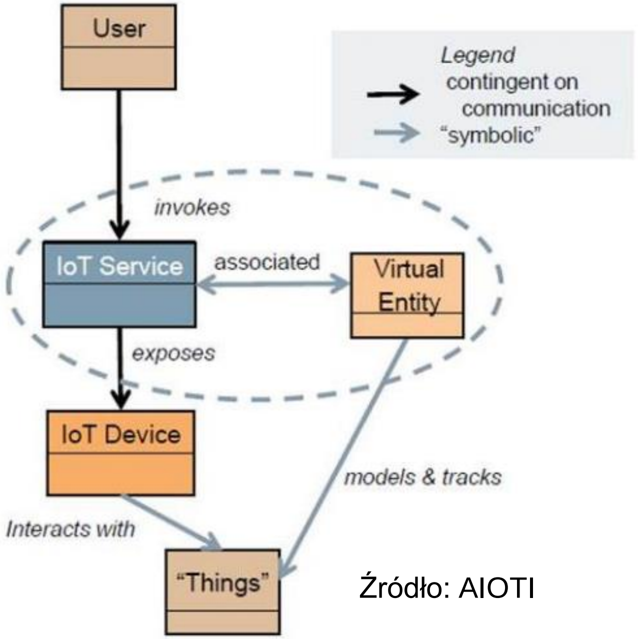
- You can derive general formulas for probabilities in two-player Penney's ante.
  - use the above formula for the ratio of probabilities and $P_1 + P_2 = 1$
  - the formulas express the probabilities in terms of the Conway leading numbers

Magic of Conway leading numbers (3/3)

- One can also derive a formula for the expected value of the number of tosses before any of the two patterns occurs.

$$E = 2 \times \frac{C^{1,1}C^{2,2} - C^{1,2}C^{2,1}}{C^{1,1} - C^{1,2} + \ C^{2,2} - C^{2,1}}$$

Our objective: distributed Penney’s ante



Źródło: AIOTI

OUR OBJECTIVE

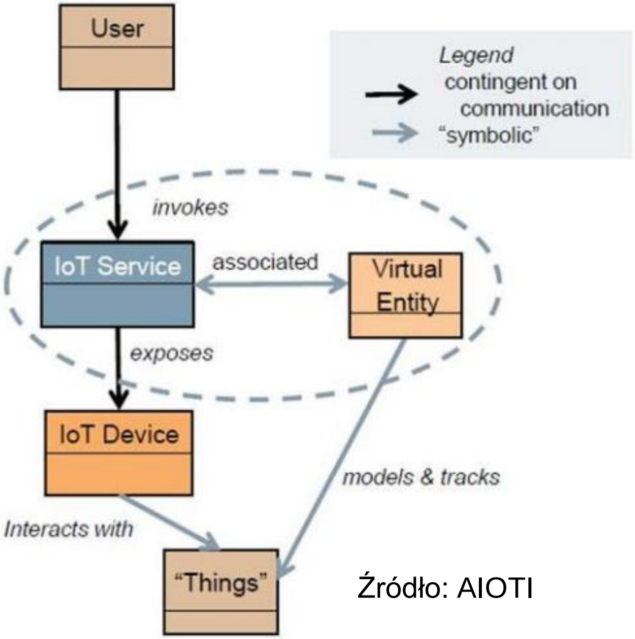
- We’ll make a distributed application that simulates the Penney’s ante game.
  - not much computation, but a lot **communication and coordination**

- For a given set of patterns, we’ll play the game multiple times.

- Major results from each game played:
  - which pattern is the winner
  - how many tosses were needed for the winning pattern to occur

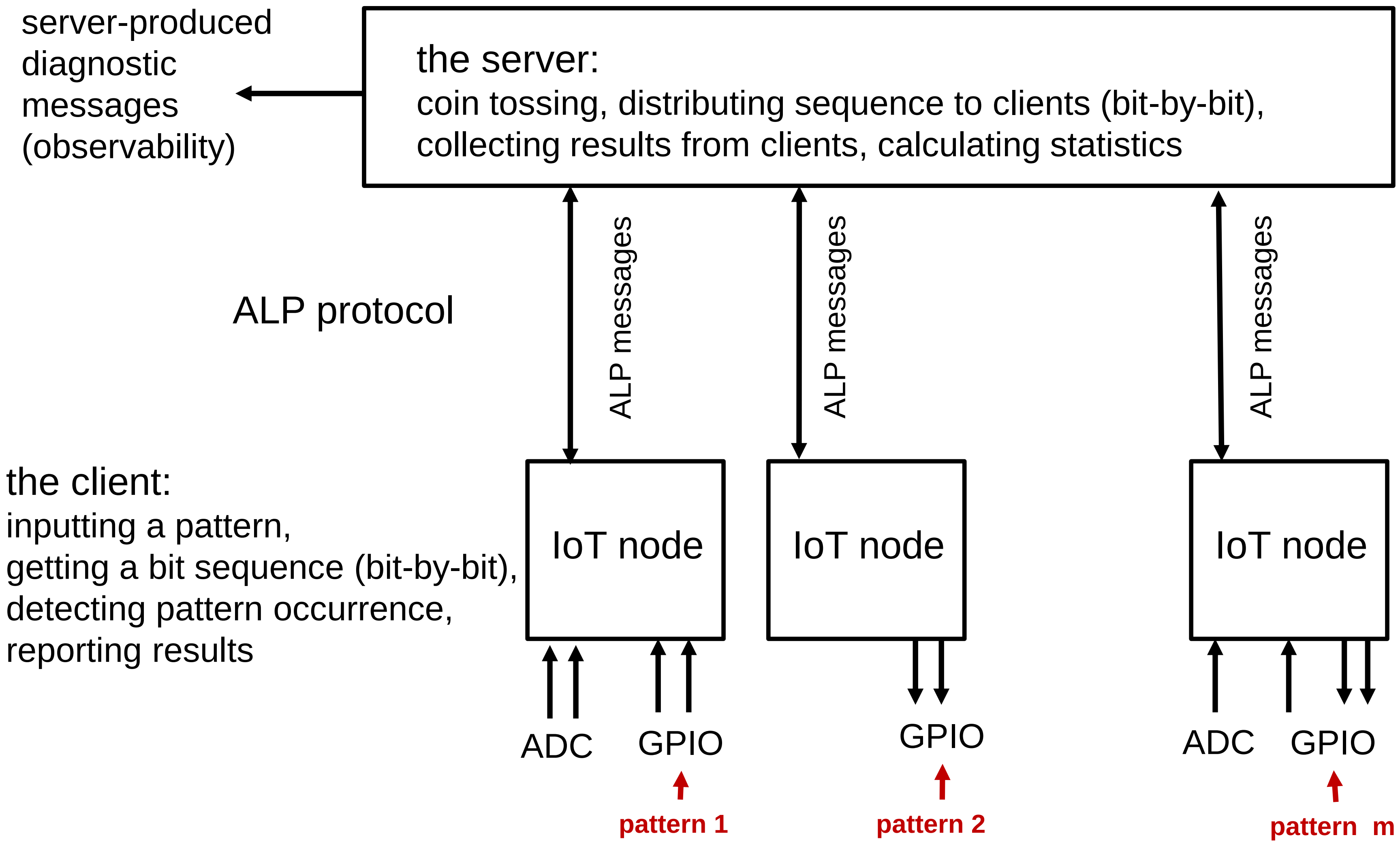
- Major results from multiple games (played with same patterns):
  - for each pattern, (an estimate of) the **probability that the pattern wins**
  - (an estimate of) the **average number of tosses** needed for a winning pattern to occur

System high-level architecture



Źródło: AIOTI

System architecture



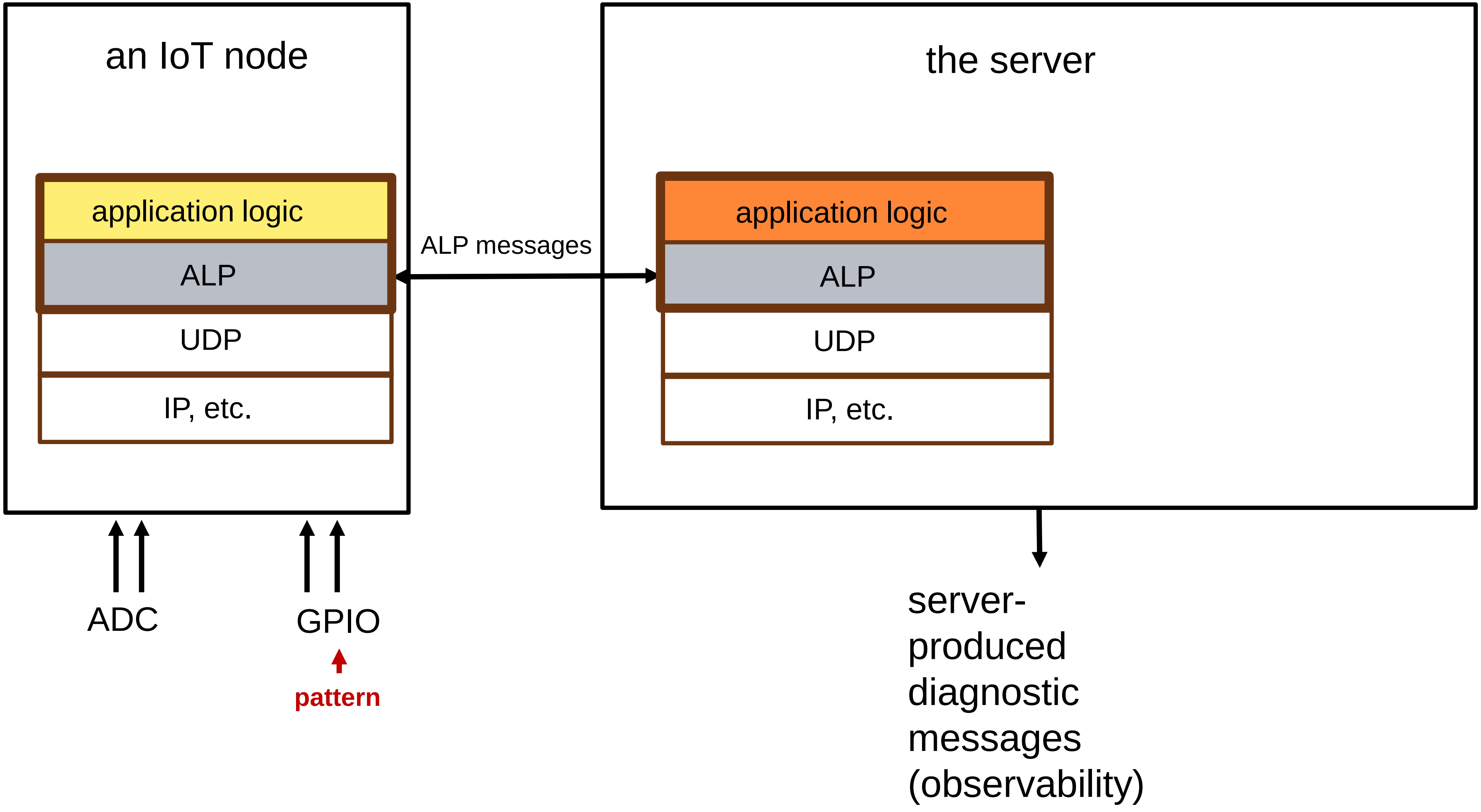
server-produced diagnostic messages (observability)

the server:
coin tossing, distributing sequence to clients (bit-by-bit), collecting results from clients, calculating statistics

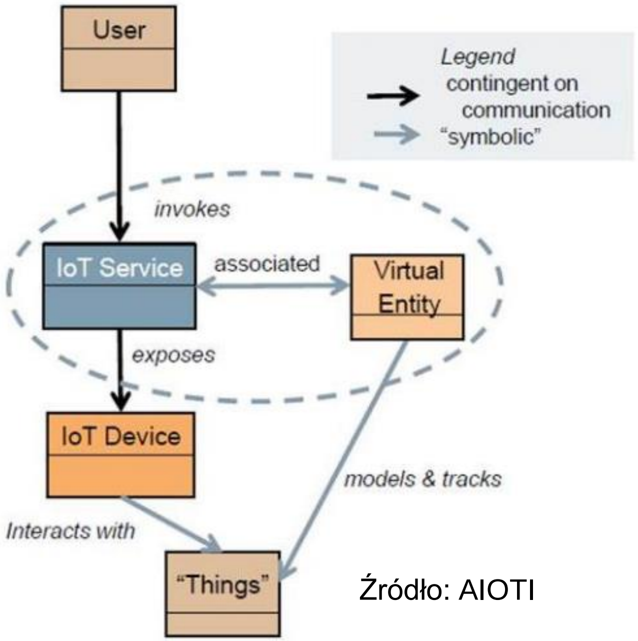
ALP protocol

the client:
inputting a pattern,
getting a bit sequence (bit-by-bit),
detecting pattern occurrence,
reporting results

SYSTEM ARCHITECTURE



ALP protocol



Źródło: AIOTI

ALP

- An application layer protocol.
- Used for communication between a client and a server.
- It’s entirely your creation.
- It should be language and platform independent.
  - e.g., clients based on different platforms, written in different languages may participate in the distributed application
  - btw, this is nothing new: making it possible for participating parties to be heterogeneous is the benefit of having a well specified protocol
    (and one of the main reasons to specify it)

ALP IS A BINARY PROTOCOL (1/2)

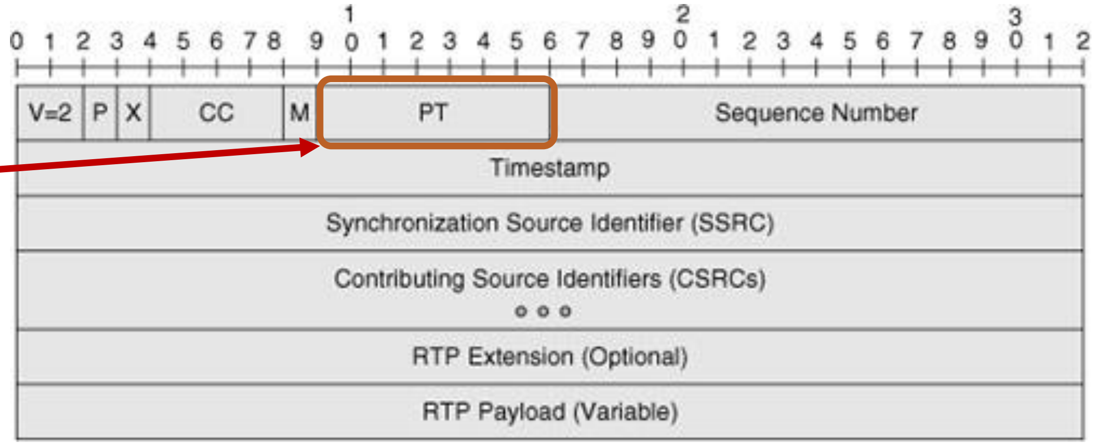
- „zaprzyjaźnij się z bajtem” (*befriend the byte*)
- ALP should be a *binary* protocol
- why? to save memory and the amount of transmitted data
- you need to define message formats at the bit level
  - identify bit fields within words
  - for each bit field, specify encodings (meanings of different bit patterns)

ALP IS A BINARY PROTOCOL (2/2)

- example: RTP (Real-time Transport Protocol)
  - a binary application layer protocol to transfer, e.g., voice in „phone calls” (VoIP)

bit number

bit field
Payload Type (PT):
a 7-bit field that identifies
the format of the RTP payload



- you may also look at RFC 791 (Fig. 4) or RFC 793 (Fig. 3)
- hint: when working with a binary protocol, make sure you understand things like:
  - network byte order
  - endianness

ALP IS A BINARY PROTOCOL (3/3)

document the format
of ALP messages like this

- another example: IPv4 (RFC 791)

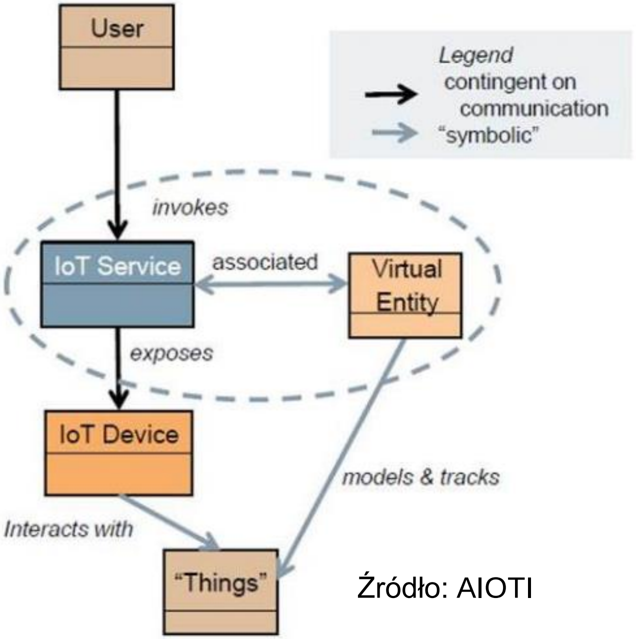
```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Version|  IHL  |Type of Service|          Total Length         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Identification        |Flags|      Fragment Offset    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Time to Live |    Protocol   |         Header Checksum       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Source Address                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Destination Address                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Options                    |    Padding    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                    Example Internet Datagram Header

                               Figure 4.
```

- in the case of ALP, you also need to specify the format of the payload
- you may also look at RFC 793 (Fig. 3)

ALP SHOULD RUN ON TOP OF UDP

- UDP often used in IoT due to its simplicity
- unreliable
  - add a simple reliability scheme
  - transmit, wait for ACK, if no ACK, retransmit

Server



Źródło: AIOTI

SERVER FUNCTIONALITY 1

- Collects registrations from participating nodes.
  - this way finds out how many nodes will play

Server functionality 2

- In a loop, manages a given number of Penney’s ante games:

```
for(i=0; i<num_games; i++)
    make_one_game();
```

SERVER FUNCTIONALITY 3

- One game looks as follows:

```
make_one_game() {

    start a game by sending a start message to all the nodes;

    do {
        toss a coin (use a random number generator);
        // the coin should be fair (both probabilities equal to ½)
        distribute H or T (one bit) to all the nodes;
    }
    while (no node reports that its pattern has occurred);

    // due to the lack of synchronization, reports from multiple nodes may arrive
    // … but only one node is the winner …
    // … namely the one that reports the least number of tosses
    // some reports (e.g., the winning one) may be delayed with respect to others
    // make sure you receive all reports

    make a record for the game just ended;
    // record which pattern won
    // record how many tosses were needed for the winning pattern to occur
}
```

Server functionality 4

- Once all the games are played, calculate estimates:
  - for each pattern, the probability that the pattern wins
  - the average number of tosses needed for a winning pattern to occur

- (It suffices to take averages.)

- Display info about the games:
  - the number of players
  - the length of the patterns (*n*)
  - how many times the game was played
  - the probabilities of winning (estimate)
  - the average number of tosses (estimate)

SERVER FUNCTIONALITY 5

- Important: do not re-initialize your random number generator after each game.
  - otherwise, you may end up playing with the same tossing history many times

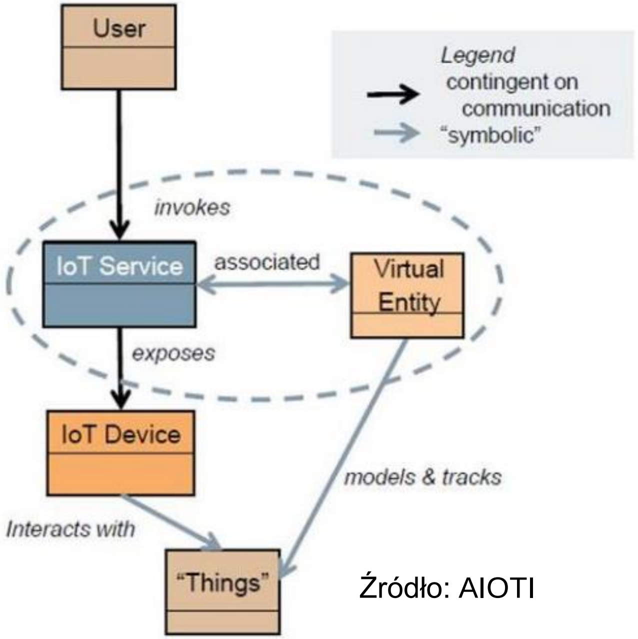
SERVER FUNCTIONALITY 6

- While doing all of the above, keep updating metrics and produce diagnostic messages.
  - examples:
    - the number of all messages sent and received so far
    - average message length
    - the number of completed games
    - …

SERVER PLATFORM

- Use Linux system API and C language facilities only
  - no libraries, no platforms, no middleware, …

Nodes



Źródło: AIOTI

NODE FUNCTIONALITY

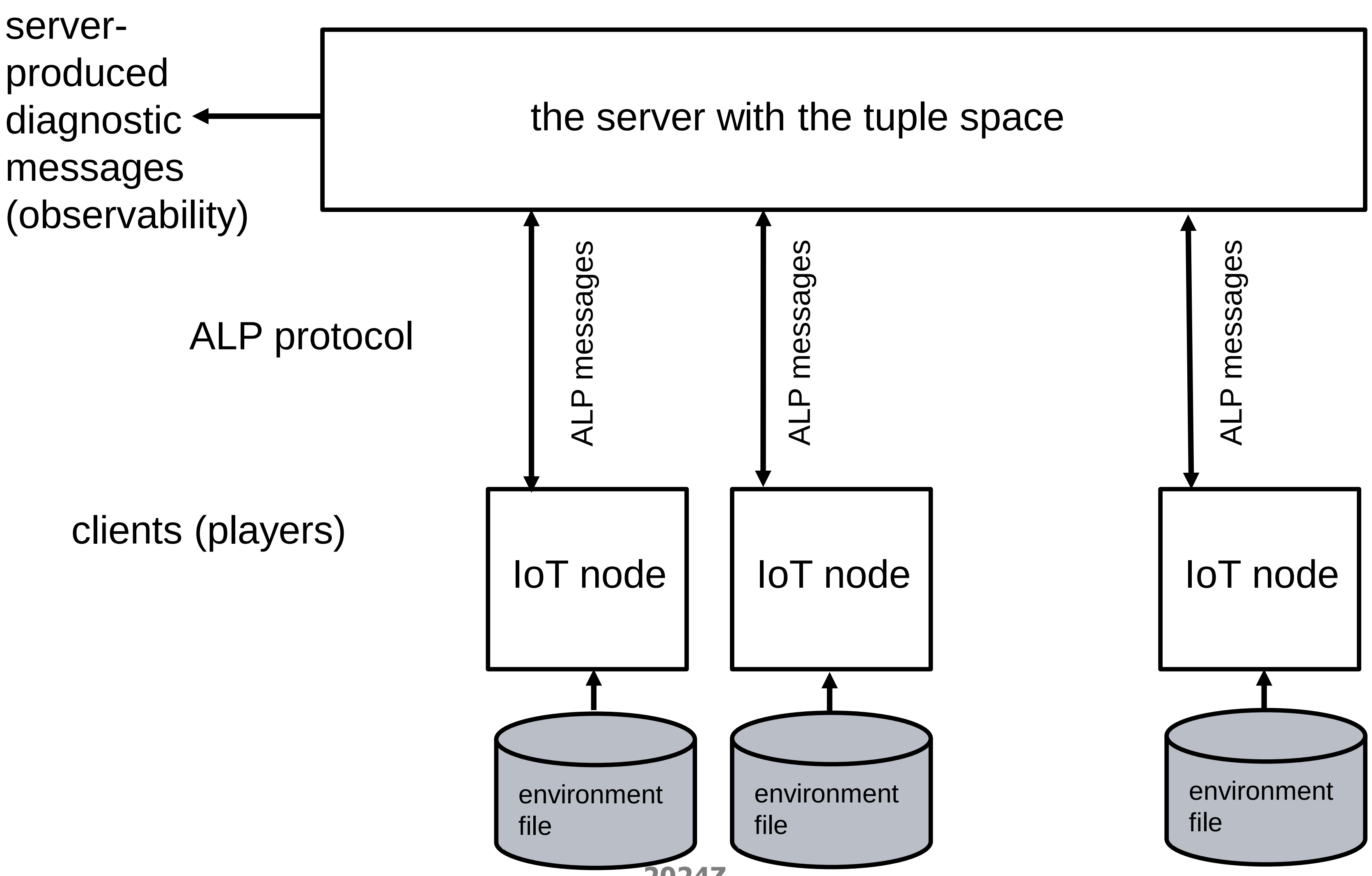
- Reads the node's pattern, using GPIO.
- Registers with the server.
- In a loop, plays Penney's ante games, as directed by the server:

```
do {
    wait_for_message;
    if(start) {
        clear_tossing_history;//important! each new game should start from scratch
        continue;
    }

    // make sure the message carries the result of a toss (H or T)
    update tossing history;
    if(pattern has occurred)
        send a report to the server
    }
} while (1);
```

NODE PLATFORM

- Arduino emulator
- Use ”standard” Arduino API and PSIR extensions only.
- The emulator supports UDP only.

ENVIRONMENT FILES



server-produced diagnostic messages (observability)

the server with the tuple space

ALP protocol

clients (players)

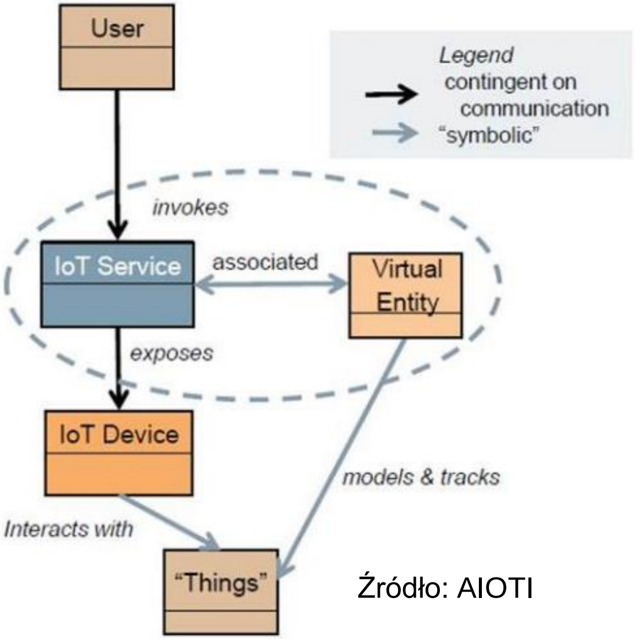
Environment file

```
+ qTemperature,quantity,Z0   # input 0=-10 1023=+20
+ qHumidity,quantity,Z1
+ sOpening,status,D1         # input 0=OPEN, 1=CLOSED
+ aSwitch,action,D2          # output 0=ON, 1=OFF


: 1000,qAirTemperature, 20
: 3000,qAirTemperature, 21
: 5000,sOpening, 0
: 7000,qHumidity, 512
```

HOW TO INPUT A PATTERN

- The length of the pattern (*n*) can be hard-coded.
- Use two GPIO pins.
- To get one bit of a pattern:
  - detect an edge on one GPIO pin
  - right after detecting the edge, read the value on the other GPIO pin
- Repeat the above until you get *n* bits.

Your results



Źródło: AIOTI

YOUR RESULTS

- source code
- a report
- a demonstration

YOUR RESULTS ARE YOURS!!!

- Freely talk with other students about concepts.
- You may reuse some items developed by another team, as long as you give credit to the team that created them.
  - this is a general rule that helps you avoid plagiarism
  - however, we reserve the right to make a judgment as to how important the reused items are, and to deduct points accordingly
  - reusing somebody else's work may turn out very costly in terms of points deducted

- We can easily see reusability!
- If we see reused items without clearly given credit, all teams with those items (including the authors) will have the same number of points deducted.
  - (we cannot investigate who has taken what from whom)

- Your best strategy, if you are cooperative: offer others advice but do not share items you are going to submit as results!

THE CONTENTS OF YOUR REPORT

1. a specification of your ALP
   - produce message formats and binary encodings for the protocol messages
     - when specifying message formats, follow a good example, e.g., RTP or IP
   - produce sequence diagrams (see the UML language to learn about sequence diagrams)
   - describe how you deal with the fact that UDP is unreliable
2. a description of your server implementation
   - components (overview of the server architecture)
   - major data structures
3. a description of your node implementation
   - include a description of your environment files
4. estimates obtained for some sets of patterns
   1. probability of winning for each pattern
   2. average number of tosses

YOUR DEMO

- prepare several nodes (players)
  - each node requires its own environment file
- during the demo
  - start your server
  - start all the players
  - …
  - wait for estimates

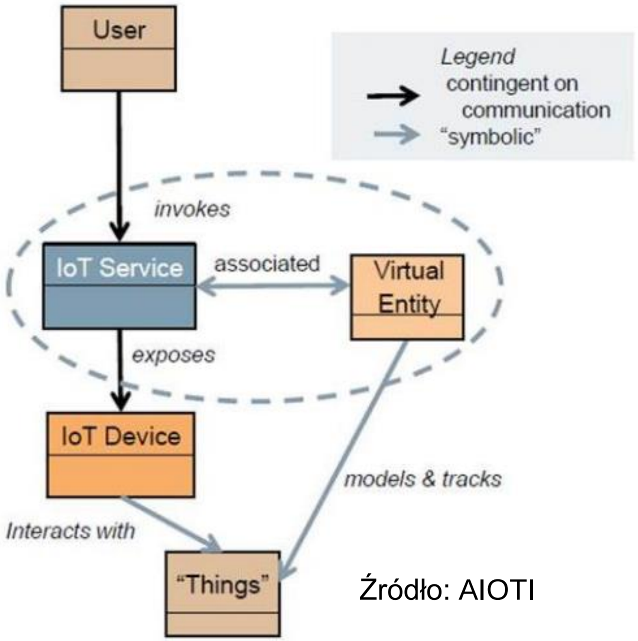
GRADING

- source code quality 10%
- report 45%
  - 1-20%, 2-10%, 3-10%, 4-5%
  - (see the contents of the report)
- demo 45%
  - what counts most is whether your software works …
  - … but whether the demo is presented smoothly counts as well

DEADLINE

- All teams must upload their results by January 20, 2025 12:00.
  - that’s Monday
- Uploading to the PSIR-supplied git account (also used for labs).

References

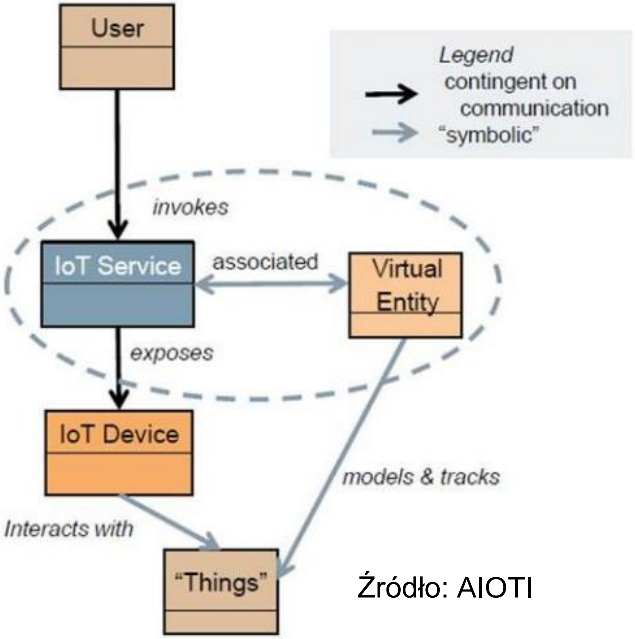


Źródło: AIOTI

References

- Isha Agarwal, Matvey Borodin, Aidan Duncan, Kaylee Ji, Tanya Khovanova, Shane Lee, Boyan Litchev, Anshul Rastogi, Garima Rastogi, Andrew Zhao *From Unequal Chance to a Coin Game Dance: Variants of Penney's Game* https://doi.org/10.48550/arXiv.2006.13002
- Stanley Collings *Coin sequence probabilities and paradoxes* Bulletin of the Institute of Mathematics and its Applications (1982) 18, 227-232

Dziękujemy za uwagę!



Źródło: AIOTI