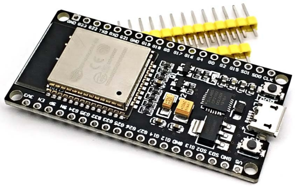
TÖL103M

Programming Projects on IoT

Autumn 2023

Programming Assignment P4



Contents:

Meeting #10 Agenda

1. Feedback on Programming Assignment P3
2. Schedule for the rest of the course
3. Group presentations
4. Programming Assignment 4
   - Setup
   - Real-time Control Protocol
   - Milestone I
   - Milestone II

1. Schedule

Schedule:

Wed 08.11:	P4 is published and introduced
	Topics for group presentations
Wed 15.11:	Milestone I of P4
Tue 21.11:	DL for P4, automated submission at Gróska
Wed 22.11:	Friendly competition (P4) and group presentations

Other:

- Remaining code reviews will be replaced by shorter) quizzes
  Take your time as some questions might be more involved

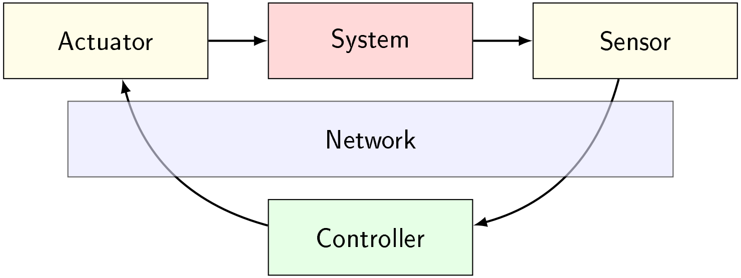
2. Group Presentations

Objective: **what's going on in the world of IoT!**

- Group presentations on freely chosen topics
- Group sizes 2-3 students (or 1 if you prefer so!)
- Topics can be also loosely related to IoT
  - Communication protocols
  - MCUs
  - Legal aspects, privacy, security
  - Home automation
  - Scripting languages LuA (from course perspective)
- Time allocation: 5-10mins per topic
- Prepare say 3 slides, present the topic, and some discussion!

Use Ed forum to find a group, or be aloud now!

3. Control systems and Time Delays



Questions

- What is the control loop
- What kind of requirements
  - e.g. on time delay for control actions (stability!)
- Some objective?

Background to P4

Grand Idea:

- Idea is to *simulate* some elements of a control system
- . . . by playing two-player tictactoe game

Mapping:

1. Objective is to win the game!
2. Control actions become *game actions*
   - That must be taken before a fixed limit
   - Node should examine different moves and choose the one that appears to be the best based on the incomplete analysis!

Programming Assignment P4

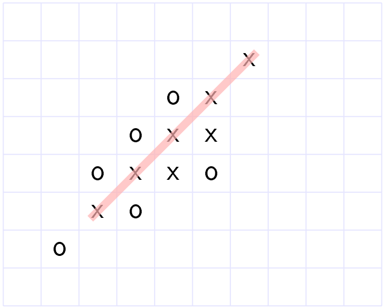
- TicTacToe will simulate a (physical) system
- Nodes tasks is to take good actions in response to state changes
  - State of the game on board
- Real-time control scenario
  - Actions must be taken in *n* seconds (e.g. 3 seconds)
  - Failure to act in time means lost game
- Master node will act as a Game server (the system)

Rules of the Game:

Tic-Tac-Toe on 30x30 board:

- Two players: player 1 has “crosses” and player 2 “nulls” (zeroes)
- The game is played on 30x30 board
- Players take turns and place one mark at a time on any free square on the board
  - Marks are in general final
  - However, the game server MAY occasionally clear some squares
- The goal is to get **five own marks in a row**
  - 4 directions: horizontal, vertical and two diagonal

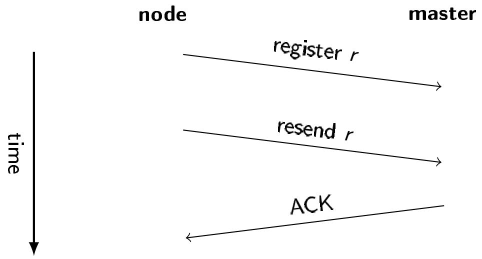
Tic-Tac-Toe



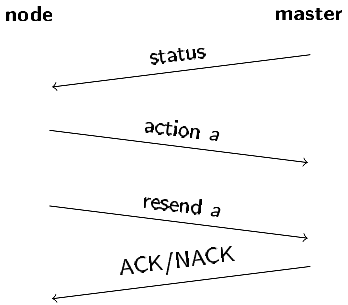
Game (Control) Protocol

- Nodes register to server to play the game
- Repeat:
  - Game status report to nodes involved
  - Node in turn responds with an action packet
- Until game is finished or k rounds have passed
- Result is communicated to everyone (+ chat channel)

Protocol



Registration process



Game actions

Source files

- Communication protocol etc. in `games.h`
- Respective routines implemented in `games.c`
- Tic-Tac-Toe specific routines are in `tictactoe.{c,h}`.

Game state in different formats:

1. Sparse internal presentation in the memory:

```
#define TICTACTOE_N      (30*30)    // number of squares
typedef struct
{
    uint8_t board[ TICTACTOE_N ];    // 0=free, 1=x and 2=o (zero)
} tictactoe_t;

uint8_t tictac_get( const tictactoe_t *b, int  i, int  j );
int     tictac_set(       tictactoe_t *b, int  i, int  j, uint8_t s );
```

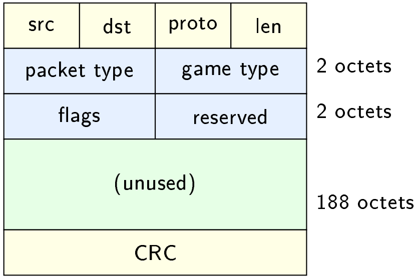
Game state takes 900 bytes

2. Dense base-3 encoding to Lownet frames uses 180 octets

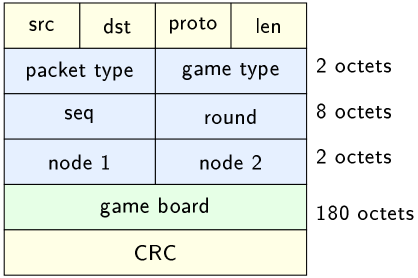
Packet types

1. **Registration**
    - `PACKET_REGISTER`
2. **Status packets**
    - `PACKET_STATUS`
    - `PACKET_WINNER_1`
    - `PACKET_WINNER_2`
    - `PACKET_TIE`
3. **Action packets**
    - `PACKET_ACTION`
    - (`PACKET_QUIT`)

Game registration and status

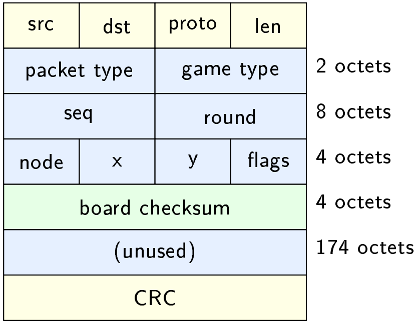


i) Register



ii) Status

Game Action packets



iii) Action

Programming Assignment P4

Milestone I: Complete the game protocol

- Skeleton code and most of the protocol is provided
- Task 1: Add checksums to some control packets

Milestone II: Implement a better strategy!

1. Actions must be taken in time!
2. Must be feasible actions
3. Implement a winning strategy against the random player:
    - Objective: win 8 games in a row

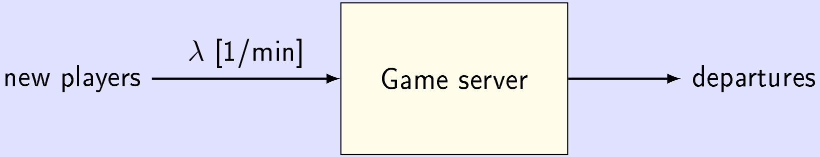
All modifications can be done in a single file: `tictac_node.c`.

Little bit of Dimensioning

- **Resource allocation** is a general problem:
    1. How large buffer should be?
    2. How fast server/processor should be? Or how many we need?
    3. How fast linkspeed we need?
- More resources leads to better performance
    1. Customers are more happy
    2. Deadlines are rarely exceeded
- It also leads to idling servers, unused resources
    1. There may be a monetary cost for it
    2. Or in resource constrained environment like MCU, we have a limited pool of memory shared by all tasks and activities
- Ideally, **all resources should be in use**, and at the same time, **system's performance should be *acceptable***.

Dimensioning is about walking on the thin line between the two contradicting objectives!

Dimensioning the Game Server

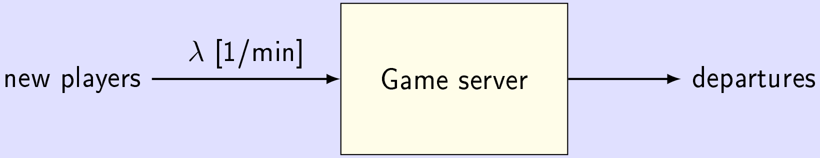


Questions:

1. How many players in the system *on average*?
2. How many parallel games in the system *on average*?
3. Fixed memory allocation, max. k games
   - How many players in the system on average?
   - How many players are rejected per day?

We need **a model** or **an experiment** to answer these questions!

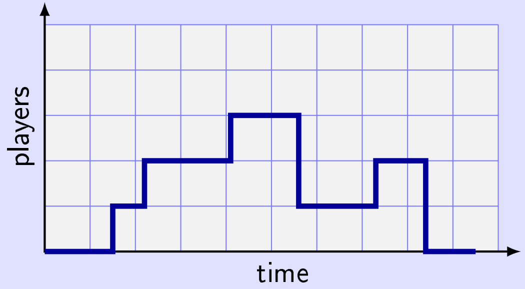
Dimensioning the Game Server



Model:

1. **Arrival process:** Suppose *inter-arrival times* between players are i.i.d., $A_i \sim A$
   - A is some known random variable with mean $1/\lambda$
2. **Pairing process:** each arriving player needs a pair
   - Players may have to **wait in a queue before an opponent is assigned**
3. **Duration of a game:** how long does each game last.
   - We model this also with i.i.d. random variables $X_i \sim X$ [second]
4. **System capacity:** Suppose we can host any number of concurrent games.

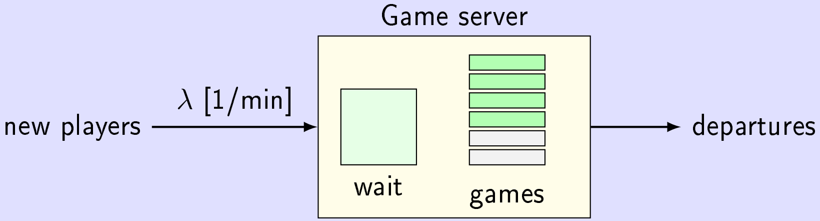
System Analysis



- We have defined a *stochastic model*
  - It evolves in time (like the real system)
  - State is described by the present players, on going games, how long they have been played, etc.
    - e.g., what is the remaining duration of a game that has already lasted 2 minutes?
- Models can be i) simulated or ii) analyzed

Earlier question: *How many players or games in the system?*

Refined Model



1. **Waiting players:**
   - Arrival rate $\lambda_w = \lambda/2$ (why?)
   - Mean waiting time (on condition one waits):

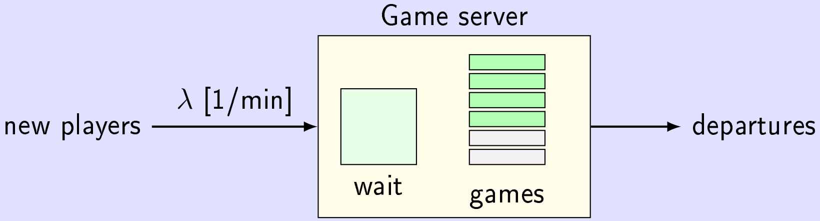
$$w = \mathrm{E}[A] = 1/\lambda$$

   - *Magic – Little's result* $N = \lambda T$

$$n_{\text{waiting}} = \lambda_w \cdot w = \frac{1}{2}.$$

The waiting position is taken half of the time!

Refined Model



2. Active games:
   - Arrival rate $\lambda_g = \lambda/2$ (why?)
   - Mean duration of a game: $x = \mathrm{E}[X]$
   - Let us invoke *Little's result* $N = \lambda T$ again $\boxed{n_{\text{games}} = \lambda_g \cdot x = \frac{\lambda x}{2}}$

On average we have $\frac{1}{2} + \lambda x$ players online!

Dimensioning questions

- We have scarce resources in our MCUs
  - Fixed memory allocations should not be excessive
  - But we do not want to reject many players either!
- Q1: How much memory should be allocated for games?
- Q2: How much memory should be allocated for players?

Common practice is to apply the so-called ***square-root staffing rule***:

$$m = n + \alpha \cdot \sqrt{n},$$

where

- n is the average number of "customers" in infinite system
- m is the size of the finite system
- α is some constant

More delicate pairing process

Ranking players:

- What if players have some ranking based on their past performance.
- For example, suppose we have classified them into three groups:
  a) advanced,
  b) intermediate and
  c) novice.

More sophisticated rule would never pair an advanced player against a novice

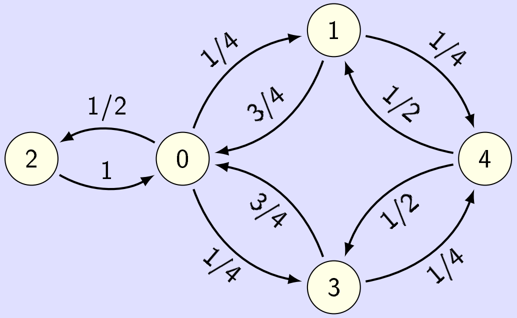
- How would this change our system?
- Q1: Can we still analyze it?
- Q2: How many players now wait, on average? Maximum number?
- Q3: What is the fraction of players who play against an opponent with same level?
- Q4: What is the probability that you meet a player with equal ranking?

Markov chain

The system state can be described by a Markov chain!

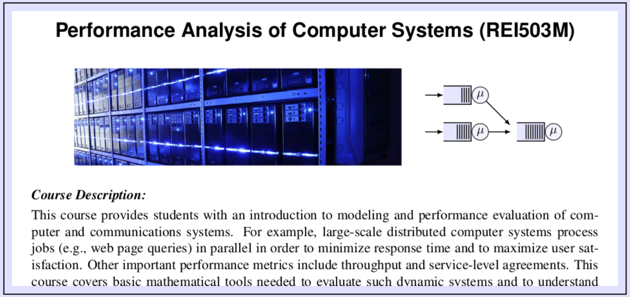
state	description
0	no one waiting
1	advanced player waiting
2	intermediate player waiting
3	novice player waiting
4	advanced and novice player waiting

Suppose new player is advanced, intermediate and novice with probabilities 1/4, 1/2 and 1/4.



This kind of questions are tackled in

REI503M Performance Analysis of Computer Systems:



Performance Analysis of Computer Systems (REI503M)

Course Description:

This course provides students with an introduction to modeling and performance evaluation of computer and communications systems. For example, large-scale distributed computer systems process jobs (e.g., web page queries) in parallel in order to minimize response time and to maximize user satisfaction. Other important performance metrics include throughput and service-level agreements. This course covers basic mathematical tools needed to evaluate such dynamic systems and to understand