iSLIP ReadMe

Handed by:

Name: Amit Davidi	ID: 208630103
Name: Guy Davidi	ID: 205493448

Computer communications
TAU Faculty of Engineering
2nd semester, 2022-2023

iSLIP Packet Scheduler

This code implements an iSLIP packet scheduler. The iSLIP algorithm is a widely used algorithm for scheduling packet transmissions in high-speed networks. The code takes input packets, schedules them using the iSLIP algorithm, and outputs the scheduled packets.

Usage

The compiled program can be run with the following command:

```
Input_packets.txt > ./islip.exe <N> <k> <r>
```

- <N>: The number of ports in the network.
- <k>: The number of iterations the iSLIP algorithm will run.
- <r>: The run ID, used to generate a log file (<r>.log.txt) that contains the state of the buffers at each time step.
- <input_packets.txt>: A file containing the input packets, where each line represents a packet with the format <time> <arrivalPort> <destinationPort>. The packets should be sorted by their arrival time.

Algorithm

The code implements the iSLIP algorithm as follows:

1. Initialize data structures and allocate memory.
2. Read packets from the input and enqueue them in the appropriate queues based on their arrival and destination ports.
3. For each time step: a. Execute the iSLIP algorithm for k iterations. b. Log the state of the buffers in the log file. c. Move to the next time step.
4. Send the remaining packets in the queues. (after done reading the file)
5. Free allocated memory and close the log file.

The iSLIP algorithm consists of three stages: Request, Grant, and Accept. The algorithm iterates k times, and in each iteration, it performs the following steps:

1. Request Stage: Each input port sends requests to the corresponding output ports for available packets.
2. Grant Stage: Each output port grants a request to one of the input ports based on a round-robin selection. If an output port hasn't established a connection, it selects an input port that has requested it.
3. Accept Stage: Each input port accepts a grant from one of the output ports based on a round-robin selection. If an input port hasn't established a connection, it selects an output port that has granted it.

Once a match is found, the input and output ports are marked as matched, and the corresponding packet is sent.

Data Structures

The code uses the following data structures:

- Packet: Represents a packet with a time, arrival port, destination port, and a pointer to the next packet.
- Queue: Represents a queue of packets, consisting of a pointer to the first and last packets and the size of the queue.

Functions

The code includes several functions:

- isEmpty(Queue *Q): Checks if a queue is empty.
- Enqueue(Queue *Q, Packet *pkt): Inserts a new packet into the queue.
- Dequeue(Queue *Q): Removes and returns the first packet from the queue.
- islip(Queue **Queues, int N, int k, int r, int time_step, int *acceptRRptrs, int *grantRRptrs): Implements the iSLIP algorithm using the provided parameters and the queues.
- main(int argc, char** argv): The main function that handles program initialization, reads input packets, executes the iSLIP algorithm, and writes the log file.

Log File

The code generates a log file named <r>.log.txt, where <r> is the run ID provided as a command-line argument. The

README - Traffic Generator

This code is a simple program written in C that generates and outputs packets for a network simulation. The packets are randomly generated and assigned source and destination ports based on certain probabilities.

Code Structure

The code consists of several functions and a main function. Here's an overview of each component:

rollUniform Function

```
double rollUniform()
```

This function generates a random number between 0 and 1 using the rand function and divides it by RAND_MAX to obtain a uniform distribution.

rollPortNum Function

```
int rollPortNum(int d_flag, int N, int arrival_port)
```

This function determines the output port for a packet based on the provided parameters. If the d_flag is set (non-zero), it sets the output port to be: (otherwise – a random number from a uniform distribution)

$$\Pr[\text{destination port} = j \mid \text{arrival port} = i] = \begin{cases} 2/3 & \text{if } j = i \\ 1/3 & \text{if } j = i + 1 \bmod N \\ 0 & \text{otherwise} \end{cases}$$

main Function

```
int main(int argc, char** argv)
```

The main function is the entry point of the program. It parses command-line arguments, initializes variables, sets up the random seed, and generates and outputs packets based on the specified parameters. It also handles error checking for incorrect command-line usage.

Usage

To compile and run the code, use the following command:

```
./tr_gen.exe N T seed p [-d]
```

Replace N, T, seed, and p with the desired values for the simulation. The additional optional -d flag can be used to enable a specific behavior for calculating the output port.

Command-line Arguments

The program expects either 5 or 6 command-line arguments, as follows:

```
tr_gen.exe N T seed p [-d]
```

- N: The number of ports in the network.
- T: The number of time steps for the simulation.
- seed: The seed value for the random number generator.
- p: The probability of a packet arriving at a port.
- -d (optional): Enables a specific behavior for calculating the output port. If present, the program will use a probability-based calculation; otherwise, it will use a uniform random selection.

If the provided command-line arguments are incorrect, an error message will be displayed.

Packet Generation

The program generates packets based on the specified parameters. For each time step (t) from 1 to T, it iterates over each port (port) from 1 to N. If the randomly generated value from rollUniform is less than p, a packet is considered to have arrived at the current port. The program then calls rollPortNum to determine the destination port (dest_port) based on the probability flag and the current port.

Finally, the program outputs the packet information in the format: <time step> <source port> <destination port>.