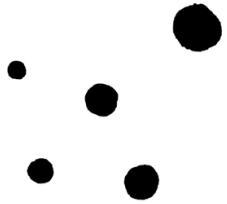


Wireless Networks Lab

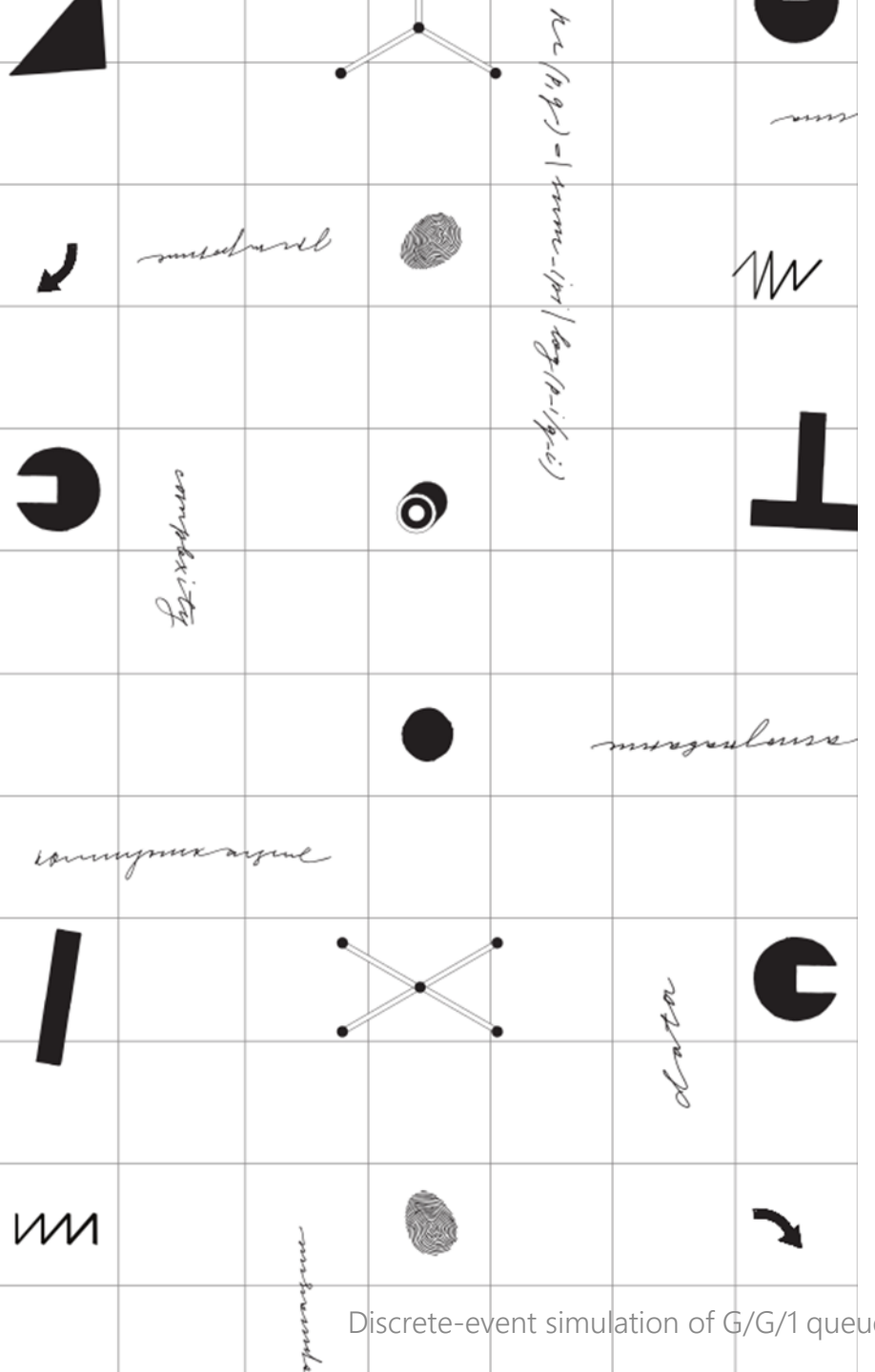
KHARKEVICH INSTITUTE FOR INFORMATION TRANSMISSION PROBLEMS
OF THE RUSSIAN ACADEMY OF SCIENCES

Discrete-event simulation of G/G/1 queueing system

Network simulation practicum

Senior researcher @ Wireless Networks Lab

Andrey Belogaev



Wireless
Networks Lab
IITP RAS

Discrete-event simulation

in C++ custom simulation platform

Events

Wireless Networks Lab
IITP RAS

Each event has the following properties:

- Unique identifier
- Moment of *model time,* in which the event happens
- Sequence of actions (function) that should be executed (called) in that moment

During the execution of the event's actions new events can be created, and/or some events can be cancelled.

Event queue

Wireless Networks Lab
IITP RAS

Event queue should be sorted in the ascending order of time moments when these events happen.

$$t1 \le t2 \le t3 \le \ldots \le tN$$

(ID#1, t1, func1)	(ID#2, t2, func2)	(ID#3, t3, func3)	...	(ID#N, tN, funcN)

```
func1 (args) { ...
    if <condition 1> {
        create new event (ID#M, tM, funcM)
        delete event (ID#J)
    }
...}
```

Data structure?

Event queue

Wireless Networks Lab
IITP RAS

Event queue should be sorted in the ascending order of time moments when these events happen.

$$t1 \le t2 \le t3 \le \ldots \le tN$$

(ID#1, t1, func1)	(ID#2, t2, func2)	(ID#3, t3, func3)	...	(ID#N, tN, funcN)

```
func1 (args) { ...
    if <condition 1> {
        create new event (ID#M, tM, funcM)
        delete event (ID#J)
    }
...}
```

Data structure?

std::map<t, event>

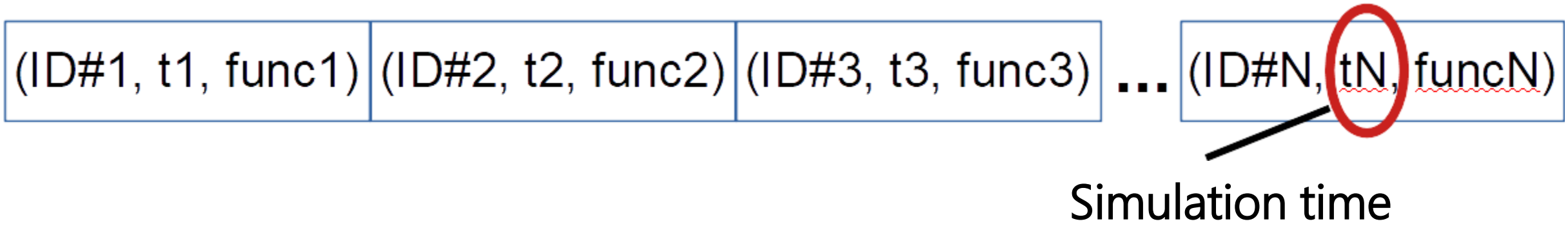
Terminating event

Wireless Networks Lab
IITP RAS

The last event that finishes the simulation.

After this event:

- all remaining events in the queue are deleted;
- the memory allocated for different data structures/objects is freed;
- calculation and saving of output statistics is done.



Callbacks

Wireless Networks Lab
IITP RAS

Description: server sends packets every *gap* seconds.

Class **Simulator**

Methods:

ID Schedule (t, func)

Fields:

std::map<t, func> queue

Class **Server**

Methods:

```
Send () {
    send packet;
    Schedule (gap, Send);
}
```

Fields:

Time gap

Class Simulator does not have any field of type Server. How to call function Send?

Callbacks

Wireless Networks Lab
IITP RAS

Description: server sends packets every *gap* seconds.

Class **Simulator**

Methods:

ID Schedule (t, func)

Fields:

std::map<t, func> queue

Class **Server**

Methods:

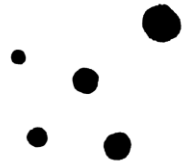
```
Send () {
    send packet;
    Schedule (gap, Send);
}
```

Fields:

Time gap

std::function<void ()> func = std::bind (&Server::Send, this)

Callbacks



Wireless Networks Lab
IITP RAS

Passing arguments to a function:

Class **Server**

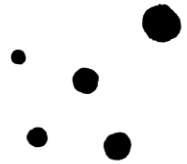
std::function<void ()> func = std::bind (&Server::Send, this, *packet*)

Simulator::Schedule (gap, func); //this is a static function (!)

Class **Simulator**

```
while (!queue.empty && event is not terminating) {
    auto func = queue.begin () → second;
    func (); //packet is already passed to the function with std::bind
    queue.erase (queue.begin ());
}
```

Random variables



Wireless Networks Lab
IITP RAS

An important property of the scientific experiment is its reproducibility, i.e., the ability to reproduce the same results after the second execution of the same code.

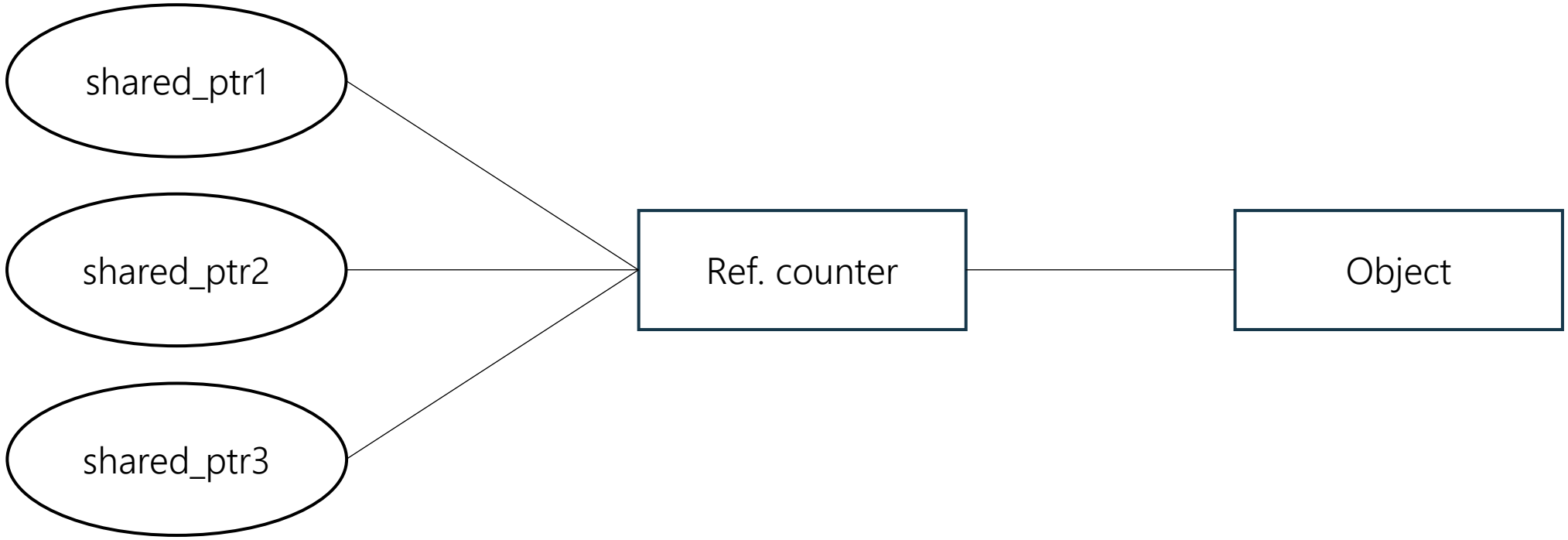
Pseudorandom numbers sequence is a sequence of generated by an algorithm numbers, that are statistically close to numbers generated from uniform distribution. From this sequence one can generate numbers from any other distribution (exponential, gaussian, etc.).

```
std::mt19937 engine (seed); // seed – number that defines the sequence
std::uniform_int_distribution<int> num (0, 99);
std::cout << “Hello, ” << num(engine) << “ world!” << std::endl;
```

“Smart” pointers

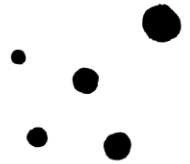
Wireless Networks Lab
IITP RAS

std::shared_ptr<Object> obj (new Object ());



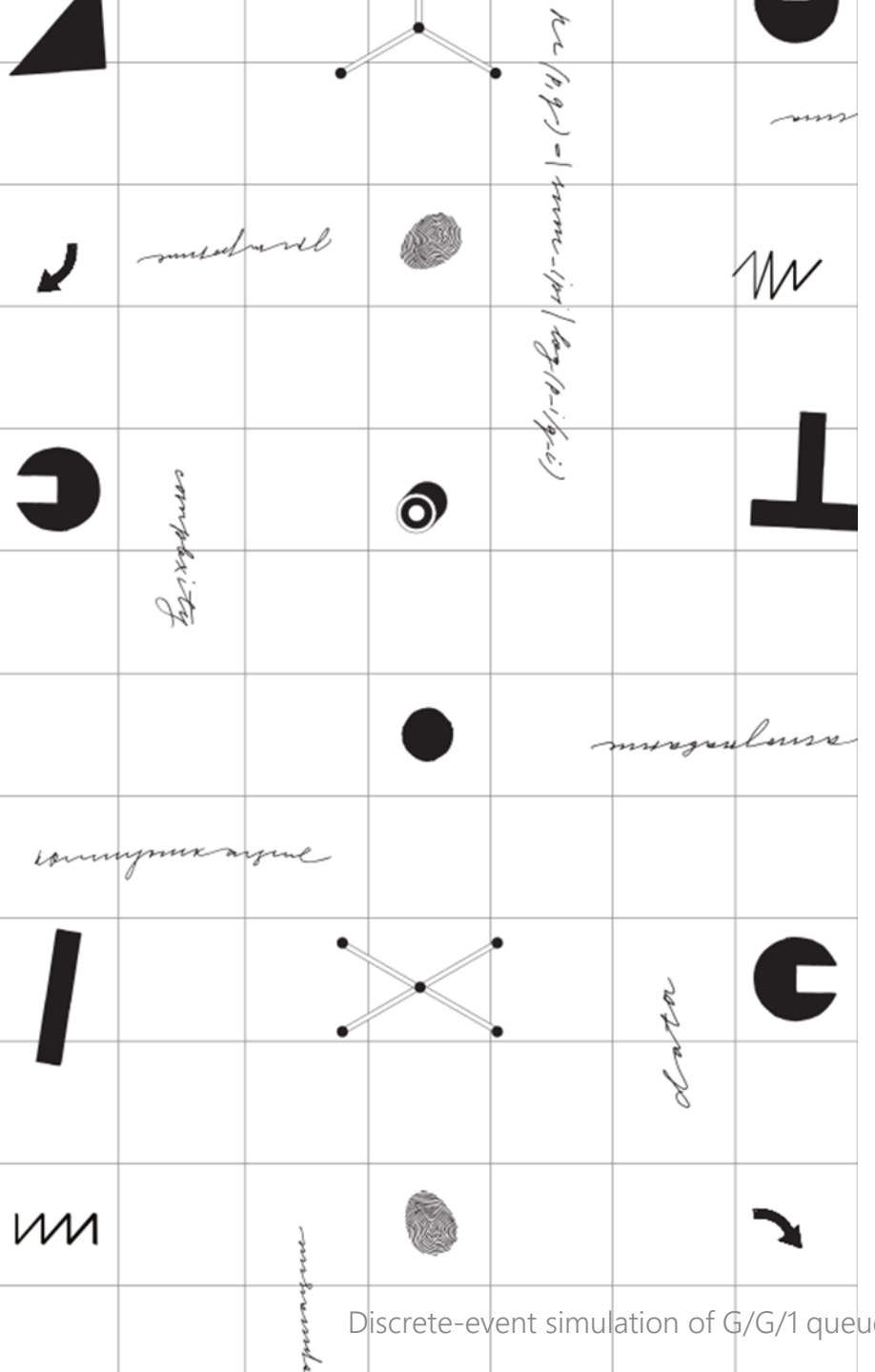
When reference counter becomes 0, the object is automatically deleted.

Scenario (main file)



Wireless Networks Lab
IITP RAS

```
int main () {
  …
  Simulator sim;
  sim.Schedule (initial events);
  sim.Schedule (terminating event);
  sim.SetSeed (seed); // seed of pseudorandom numbers generator
  sim.Run ();
  …
}
```



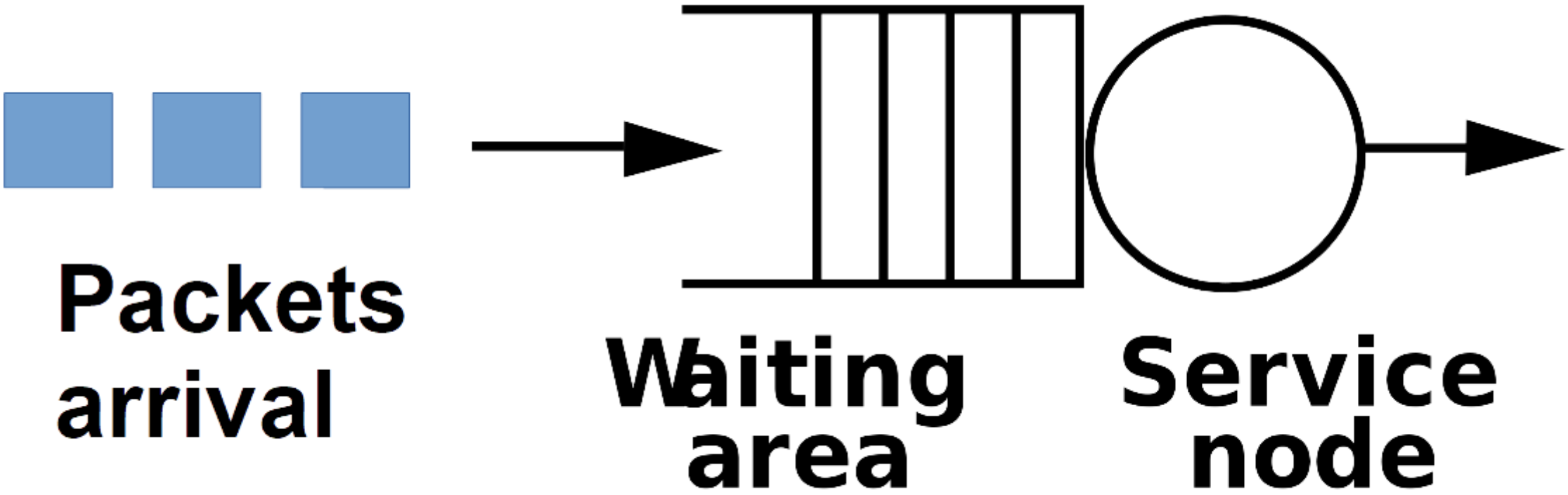
Wireless Networks Lab
IITP RAS

Modeling of G/G/1 queueing system

G/G/1 queueing system

Wireless Networks Lab
IITP RAS

Inter-packet interval and service time are random variables following some arbitrary (General) distributions.

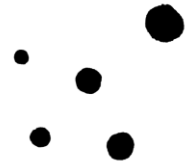


Class Packet

Wireless Networks Lab
IITP RAS

```
class Packet {
  ...
  Time arrivalTime; // time when the packet appears
  Time serviceTime; // time needed to process the packet
  ...
};
```

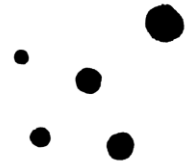
Class PacketGenerator



Wireless
Networks Lab
IITP RAS

```
template <typename distr1, typename distr2>
class PacketGenerator {…
  void Start (); // for transmission of the first packet
  void NewPacket ();
  distr1 arrival; // for inter-packet interval
  distr2 service; // for service time
  std::shared_ptr<Server> m_server;
};
```

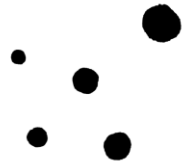
Class PacketGenerator



Wireless Networks Lab
IITP RAS

```
template <typename distr1, typename distr2>
void PacketGenerator<...>::NewPacket () {
  // create packet
  std::shared_ptr<Packet> p (new Packet (now, service () ));
  // add packet to server
  m_server → addPacket (p);
  // create callback to this same function NewPacket
  auto callback = std::bind (<...>::NewPacket, this);
  // schedule self-call after inter-packet interval
  Simulator::Schedule (arrival (), callback);
};
```

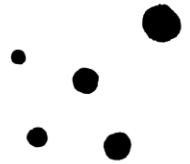
Class Server



Wireless
Networks Lab
IITP RAS

```
class Server {
  …
  void AddPacket (std::shared_ptr<Packet> packet) {
    m_queue.AddPacket (packet);
  }
  Queue m_queue;
};
```

Class Queue



Wireless Networks Lab
IITP RAS

```
class Queue {
  void AddPacket (std::shared_ptr<Packet> packet) {
    m_queue.push_back (packet);
    if (m_queue.size () == 1) {// only 1 packet in the queue
      Schedule (packet→serviceTime, RemovePacket);}
  }
  void RemovePacket () {
    m_queue.erase (m_queue.begin ());
    Schedule (m_queue.front()→serviceTime, RemovePacket);}
  std::vector<std::shared_ptr<Packet>> m_queue;
};
```

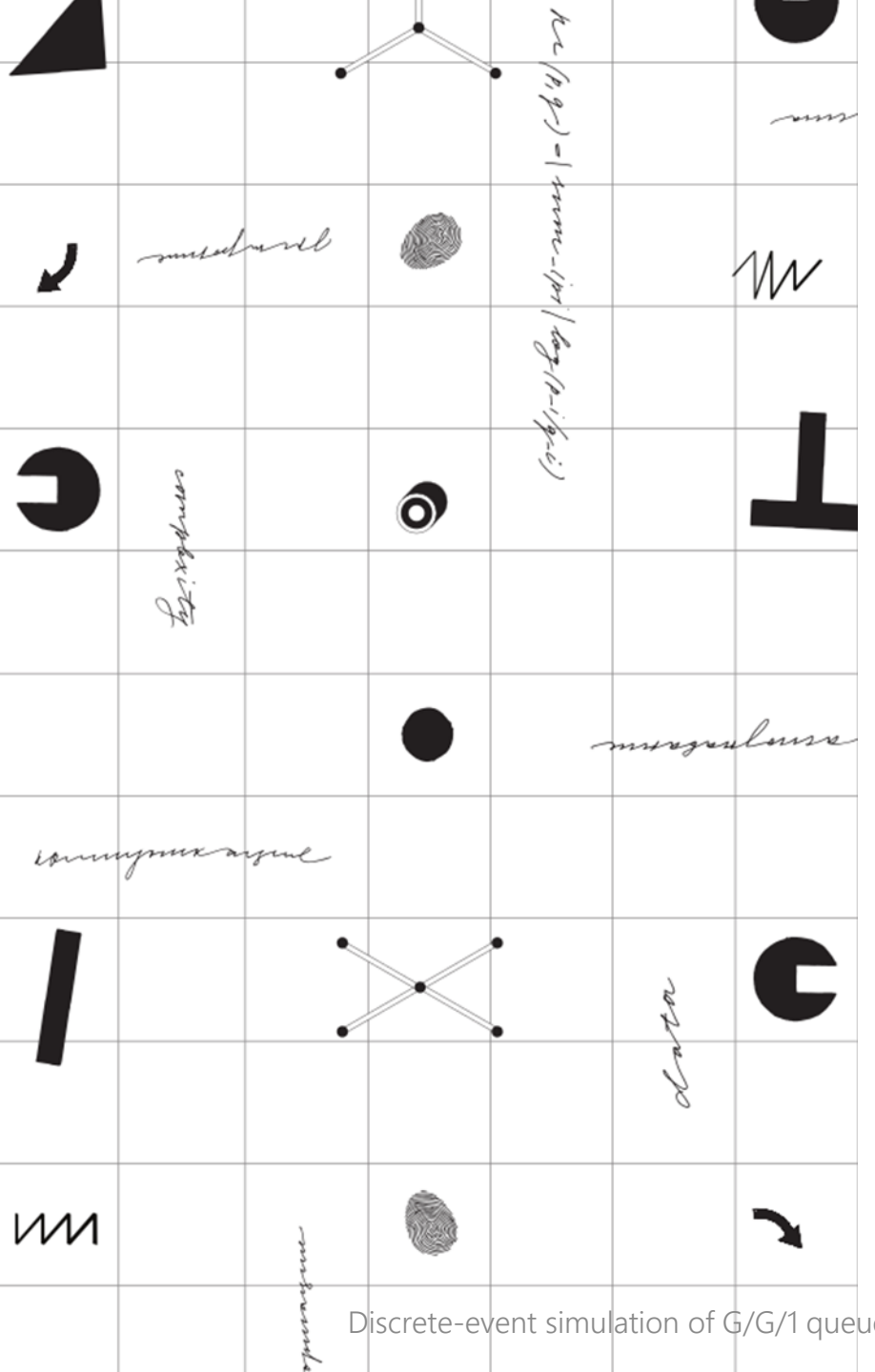
Scenario

Wireless Networks Lab
IITP RAS

```
int main () {...
  std::shared_ptr<Server> server (new Server ());
  packetGen.SetServer (server);

  Simulator sim;
  sim.SetStop (simTime);
  sim.SetSeed (seed);

  packetGen.Start ();
  sim.Run ();
}
```



Wireless Networks Lab
IITP RAS

Repo and working with code

Repository

Wireless Networks Lab
IITP RAS

Address:

sudo apt install git

git clone <address>

Build project

Wireless Networks Lab
IITP RAS

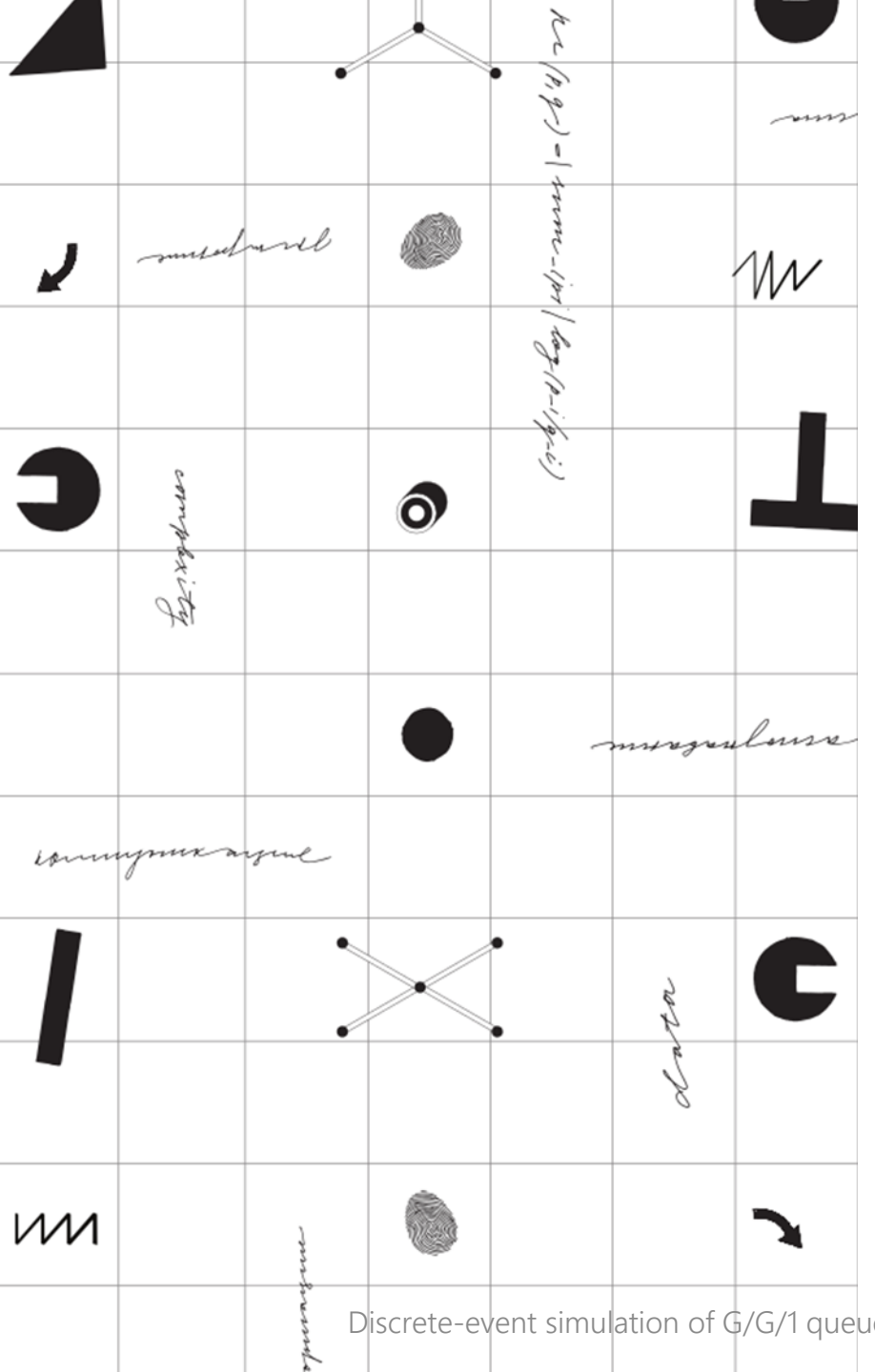
```
CXX=g++
INCLUDES=./
CXXFLAGS=-Wall -O3 -g -std=c++11

scenario : scenario.o server.o queue.o simulator.o packet.o
$(CXX) $(CXXFLAGS) -L$(INCLUDES) -o scenario scenario.o server.o
queue.o simulator.o packet.o

server.o: server.cpp server.h queue.h
$(CXX) $(CXXFLAGS) -L$(INCLUDES) -c server.cpp

clean:
rm -rf *.o scenario
```

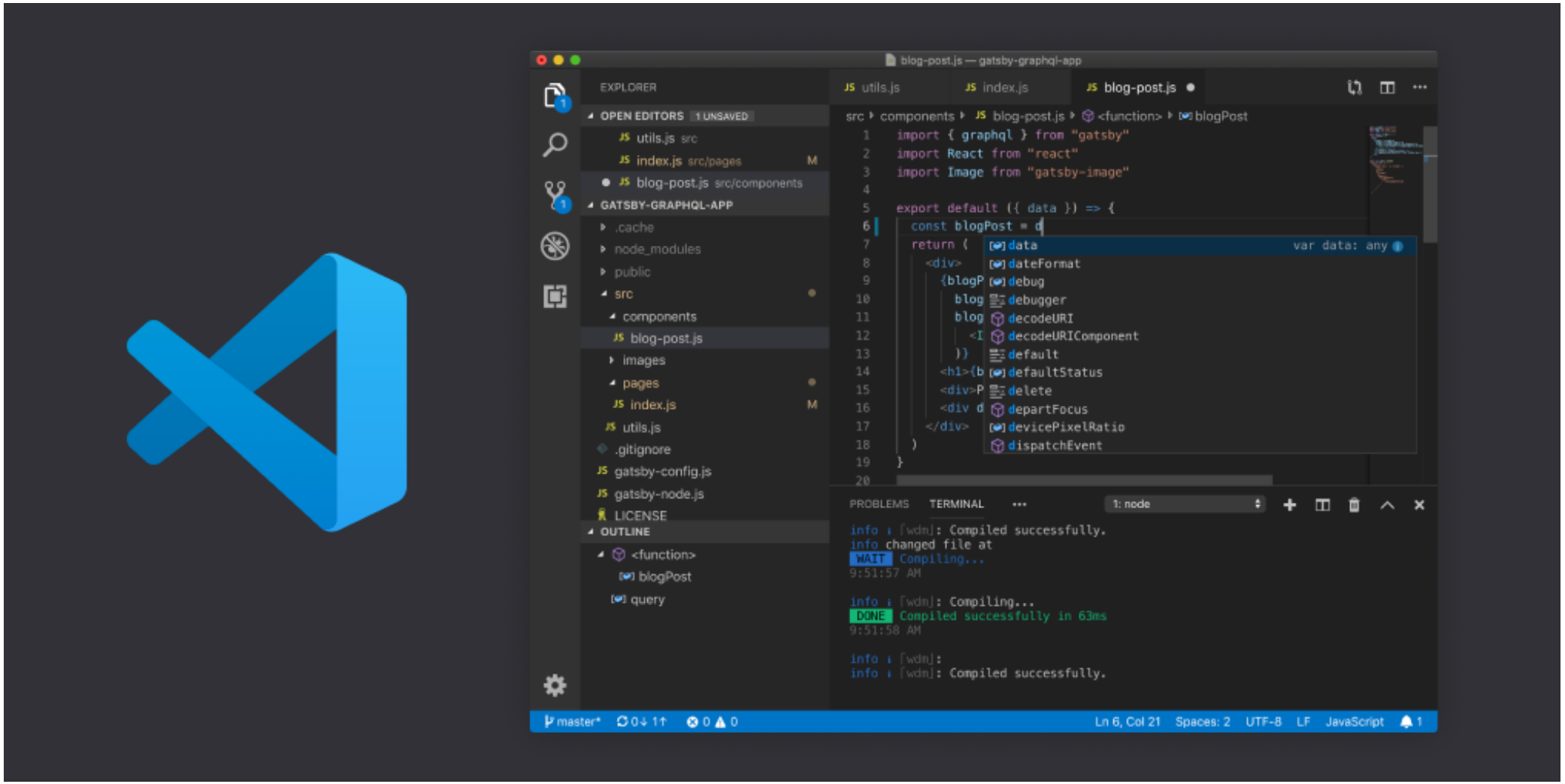
GNU Make



Wireless
Networks Lab
IITP RAS

Tools

Visual Studio Code



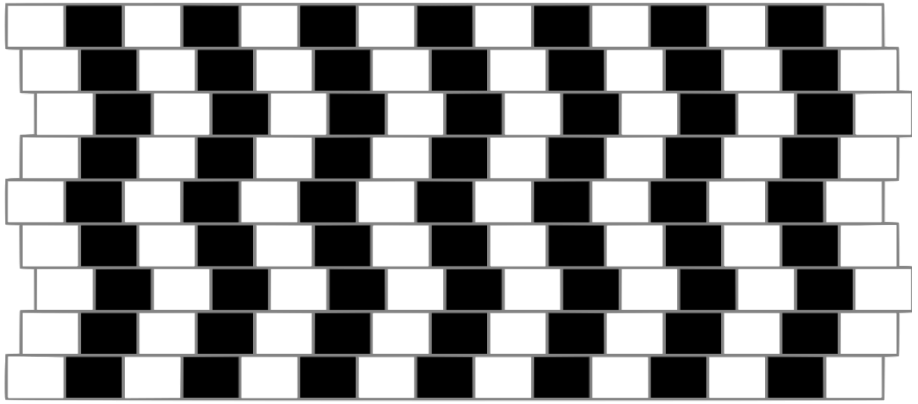
GNU Parallel

Wireless Networks Lab
IITP RAS

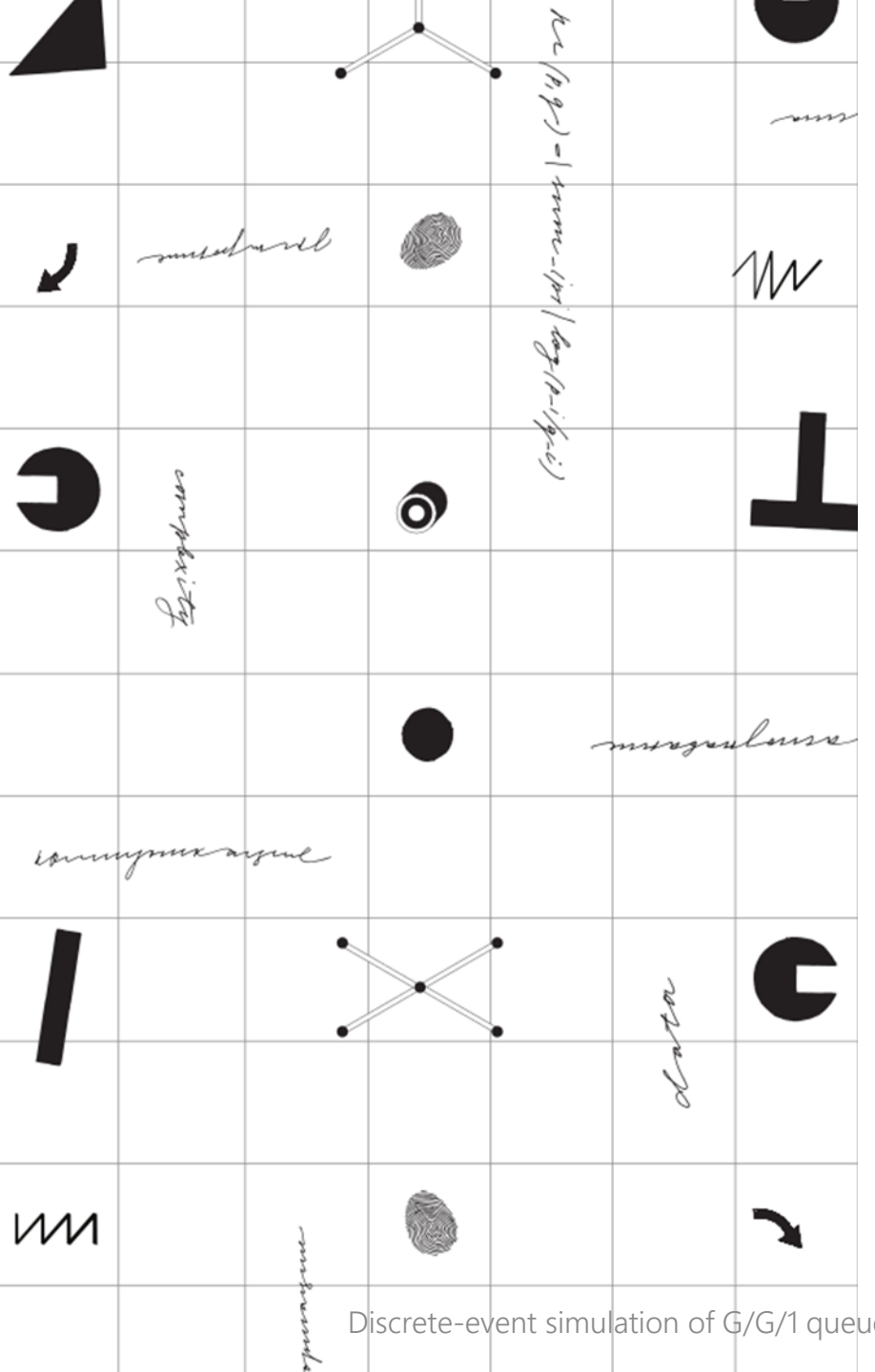
parallel ./scenario {1} {2} {3} ::: $(seq 5) ::: 1 2 3 ::: 4 5 6

To run multiple jobs simultaneously on multiple cores, instead of sequential

./scenario 1 1 4; ./scenario 2 1 4; … ; ./scenario 5 3 6



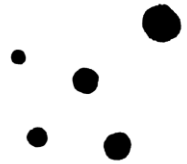
GNU parallel



Wireless
Networks Lab
IITP RAS

Task

Task description



Wireless Networks Lab
IITP RAS

1. Run experiment for queueing system M/M/1. Plot figure of mean sojourn time as a function of system load. Compare with analytical estimation.
2. Limit the size of the queue. Plot figures (analytical + simulation) for mean sojourn time and failure probability as functions of system load for different queue sizes.
3. Run experiment for M/D/1 queueing system (service time is constant) with infinite queue. Plot figure of mean sojourn time as a function of system load (analytical + simulation).
4. Prepare a report (problem statement, what you have done, description of results) in LaTeX/Word/etc.

Andrey Belogaev
a@belogaev.info
https://wireless.iitp.ru/

Wireless Networks Lab

KHARKEVICH INSTITUTE FOR INFORMATION TRANSMISSION PROBLEMS
OF THE RUSSIAN ACADEMY OF SCIENCES