

# Random Access Problem in Machine-to-Machine

- Source Code: <https://goo.gl/9zUwYw>
- **Before starting to read this article, please install chrome extension: Github with MathJax , to ensure the correctness of formula format.**
- Other requirements: `gnuplot` , `astyle`
- [Random Access Problem in Machine-to-Machine](#)
  - [Talking About Network Modeling](#)
  - [Our Environment:](#) LTE-A cellular network
  - [Part A: Access Class Barring](#)
    - [Technique Background](#)
    - [System Model](#)
    - [Simulation](#)
    - [Analysis](#)
  - [Part B: Random Access](#)
    - [Technique Background](#)
    - [System Model](#)
    - [Simulation](#)
    - [Analysis](#)
  - [Reference](#)

## Talking About Network Modeling

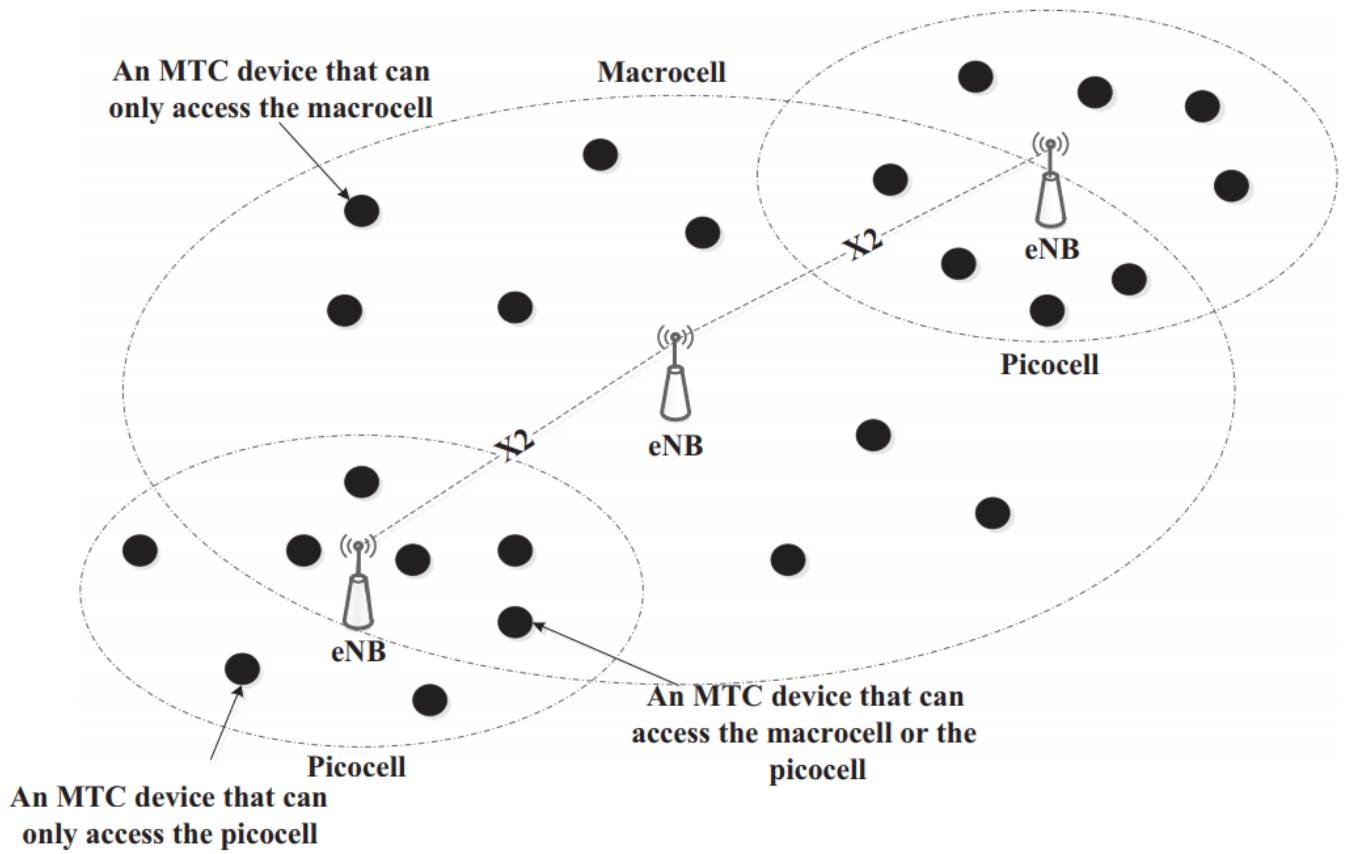
A basic way to build network model:

1. Technique background (From paper ,technical essay, magazine ... etc)
2. System model (Mathematic & Question-Defined)
3. Simulation (Base on Step 1,2 to construct program )
4. Analysis (Measure the result from program)

Base on this four method, we can easily define our question and find an efficient way to locate and find the answer.

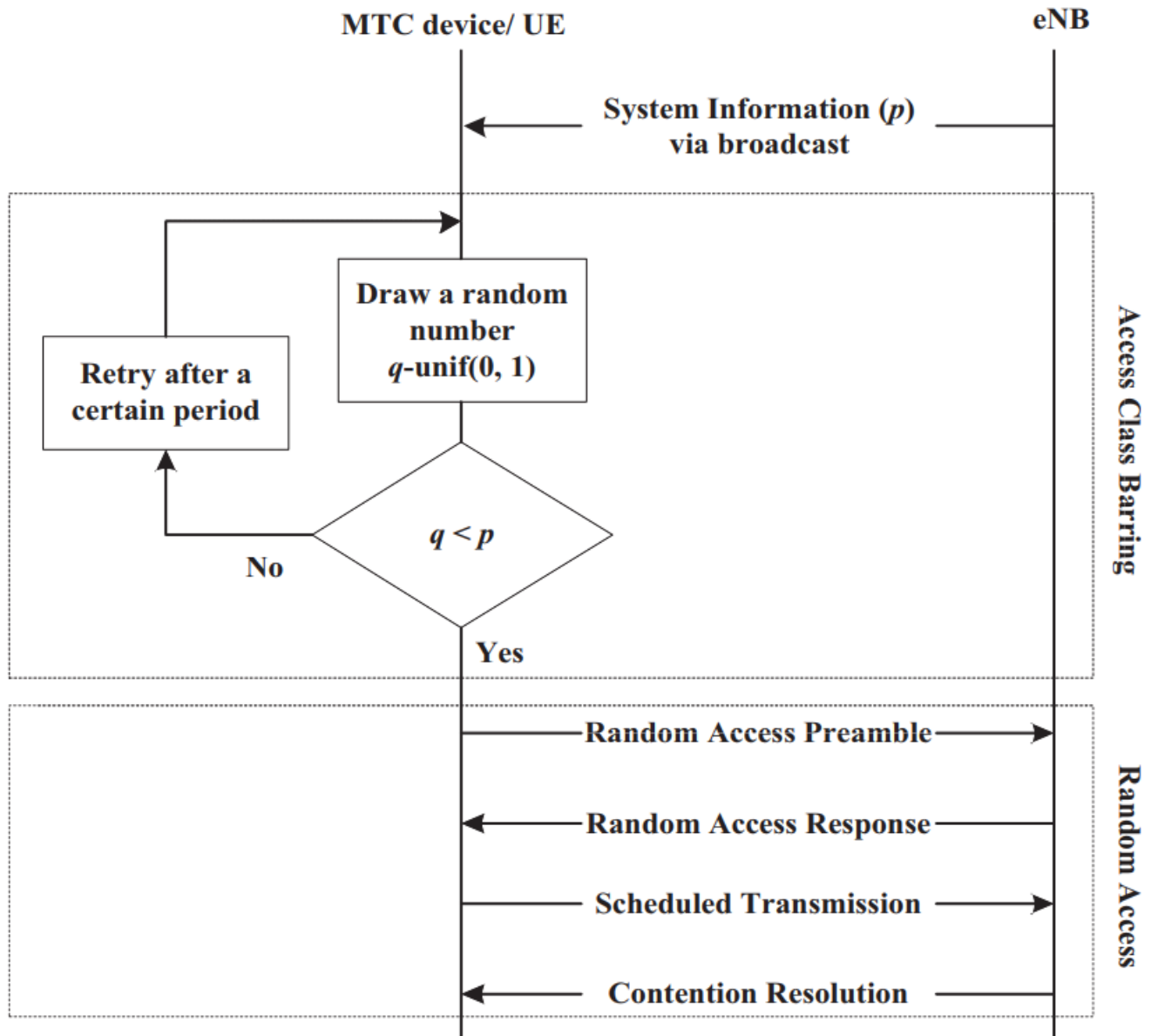
## Our Environment: LTE-A cellular network

- LTE-A cellular network [\[1\]](#)



**Fig. 1.** Architecture of an LTE-A cellular network [8].

- Access class barring procedure for an MTC device or UE [1]



**Fig. 2.** Access class barring procedure for an MTC device or UE [10].

Consider LTE-A random access procedure shown above, then we can start to construct our model - Part A (Access Class Barring) and Part B (Random Access)

## Part A: Access Class Barring

### Technique Background

- Here we can refer the **Fig 2.** to see the Access Class Barring part.
- We need to create a simulation and mathematic model to represent this part.
- Assumption was made:
  - Draw a random number  $q \sim \text{uniform}(0, 1)$  before performing the random access procedure.
  - Assume that devices are blocked for 1 ms.

- Calculate the access delay by simulation and mathematical analysis.

## System Model

Here we construct our mathematic model from technique background:

- Before calculate access delay, consider:
  - How many times do we need to complete this job? (All devices pass the access class barring)
  - Find the worst case as upperbound.
  - Then we can summation the result from best case to worse case.
- Our upperbound is generated after running the simulation model( will mention later ), we use the worse case in simulation process as our worse case in mathematic model, so that the result can plot together in well organization.
- And we using geometric distribution to calculate each expected value of each retry time, and summarize the expected value of access delay from 0 to largest key (e.g **retry times**).

## Simulation

In simulation part, I choose probability "p"(threshold) as observed value; Adjust this value to see the variation among different p value.

In each p case, we run s times of simulation routine. Each routine will finish when all devices have passed the access class barring; and each routine will record its delay times at the end of routine.

Each device will draw a random number  $q \sim \text{uniform}(0,1)$  and compare with p, if  $q < p$ , then represent the success.

After running s simulation routines, we can get an average access delay! We take this value as **access delay** of simulation.

## Analysis

Run with command `make test_a` to generate the test data of Part A.( `make test` will run all the testcase, include A and B.)

case	simulation times	<i>Number of device</i>	<i>P</i>	result
------	------------------	-------------------------	----------	--------

case	simulation times			result
1	100	30	0.1~0.9	<div><p>Access Class Barring</p><p>Figure, Simulation time=100 Number of Devices=30, interval=0.100000</p></div>
2	100	40	0.1~0.9	<div><p>Access Class Barring</p><p>Figure, Simulation time=100 Number of Devices=40, interval=0.100000</p></div>
3	10000	20	0.1~0.9	<div><p>Access Class Barring</p><p>Figure, Simulation time=10000 Number of Devices=20, interval=0.100000</p></div>
4	100000	20	0.1~0.9	<div><p>Access Class Barring</p><p>Figure, Simulation time=100000 Number of Devices=20, interval=0.100000</p></div>

As the result shown above, we can see curves of **simulation** and **mathematic** are matching with each other.

# Part B: Random Access

## Technique Background

- Here we can refer the **Fig 2.** to see the Random Access part.
- We need to create a simulation and mathematic model to represent this part.
- Assumption was made:
  - Simulate a contention-based random access procedure.
  - Collision occurs when devices are trying to transmit the same preamble.
  - There is no retransmission in your simulation.
  - Compute the success probability after the random access procedure by simulation and mathematical analysis.

## System Model

Assume there have  $M$  devices (e.g. UE / MTC device), and  $N$  preamble.

We can find the successful nodes from this paper. [\[2\]](#)

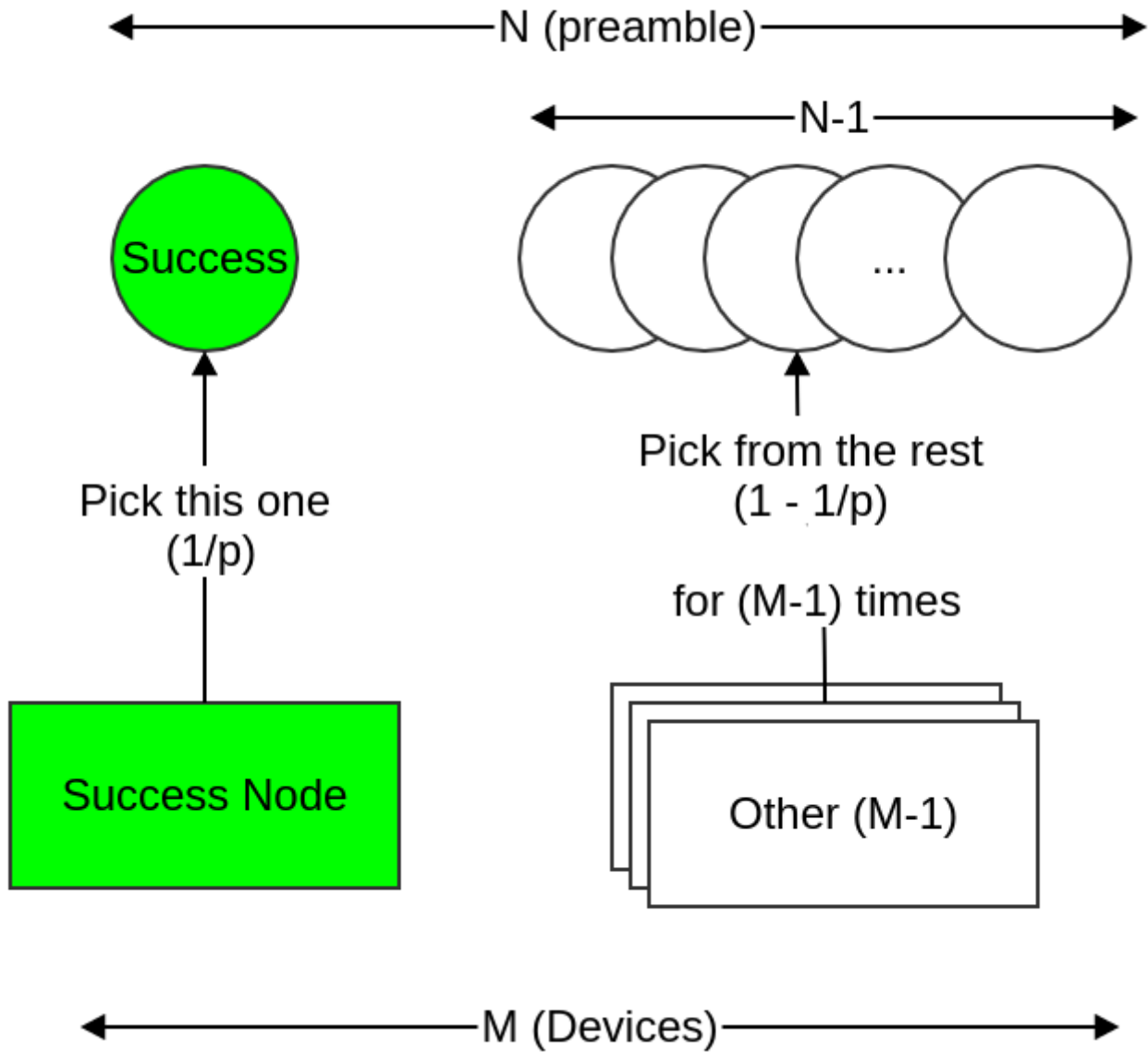
$$u(n) = n(1 - \frac{1}{p})^{n-1}$$

$n = \text{number of devices}$ ,  $p = \text{number of preamble}$

And we can easily calculate the successful probability:

$$\text{success probability} = (1 - \frac{1}{p})^{n-1}$$

This means that  $1/p$  is the preamble choose by the successful node, and the rest of nodes must choose from the rest of preamble, which is  $1 - (1/p)$  for multiply  $n-1$  times. *We don't need to care about the other one, even when number of devices is smaller than preamble*, just retain the other one won't select the preamble which chose by this successful node. We just focus on **the successful probability** of one node. And if we want to calculate the successful nodes, just multiply devices number on it.



We take this as our mathematic model.

## Simulation

In simulation part, I choose number of preamble as observed value; Adjust this value to see the variation among different preamble value with specified interval to iterate, and in each preamble case, we run  $s$  times of simulation routine.

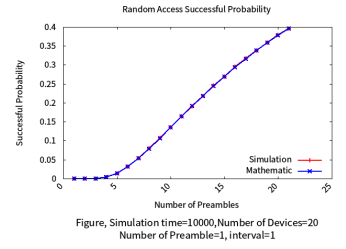
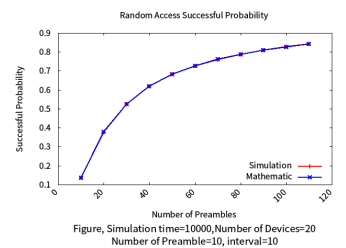
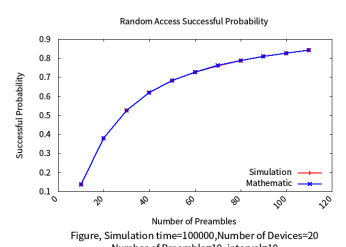
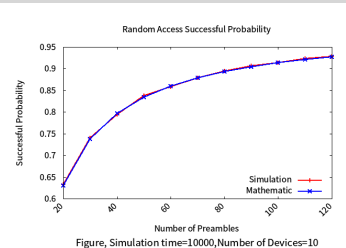
All the preamble selected by each device are generated from random number of uniform distribution. (See more in class "rand\_gen" from `utils/rand_gen.h` and `utils/rand_gen.cc`), and using `std::map` as data structure to record the selected preamble.

Consider that we don't need to care about "**retransmission**", so in each simulation routine, we only need to calculate the success devices(which using unique preamble).

And at the end of routine, using success devices to calculate **successful probability**.

## Analysis

Run with command `make test_b` to generate the test data of Part B. (make test will run all the testcase, include A and B.)

case	simulation times	Number of device	Preamble range	result
1	10000	20	1 ~ 20	 <p>Figure, Simulation time=10000, Number of Devices=20 Number of Preamble=1, interval=1</p>
2	10000	20	10 ~ 100	 <p>Figure, Simulation time=10000, Number of Devices=20 Number of Preamble=10, interval=10</p>
3	100000	20	10 ~ 100	 <p>Figure, Simulation time=100000, Number of Devices=20 Number of Preamble=10, interval=10</p>
4	10000	10	20 ~ 100	 <p>Figure, Simulation time=10000, Number of Devices=10 Number of Preamble=20, interval=10</p>

As the result shown above, we can see curves of **simulation** and **mathematic** are matching with each other.

And we can see the testcase shown above, when devices is 20, there need to use almost 120 preamble to make *successful probability* reach to **80%**.

## Reference

[1] Efficient cooperative access class barring with load balancing and traffic adaptive radio resource management for M2M communications over LTE-A. **Yi-Huai Hsu, Kuochen Wang, Yu-Chee Tseng.**  
Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan



[\[2\] Lower Bounds on the LTE-A Average Random Access Delay Under Massive M2M Arrivals.](#)

**Mehmet Koseoglu.** *IEEE, member*