**Adaptive Backoff Algorithm for Contention Window for**  
**Dense IEEE 802.11 WLANs**

<https://doi.org/10.1155/2016/8967281>

CSE 322

Computer Networks Sessional

**Name:** MD. ZARZEES UDDIN SHAH CHOWDHURY

**StudentID:** 1805009

**Lab Group**: A1

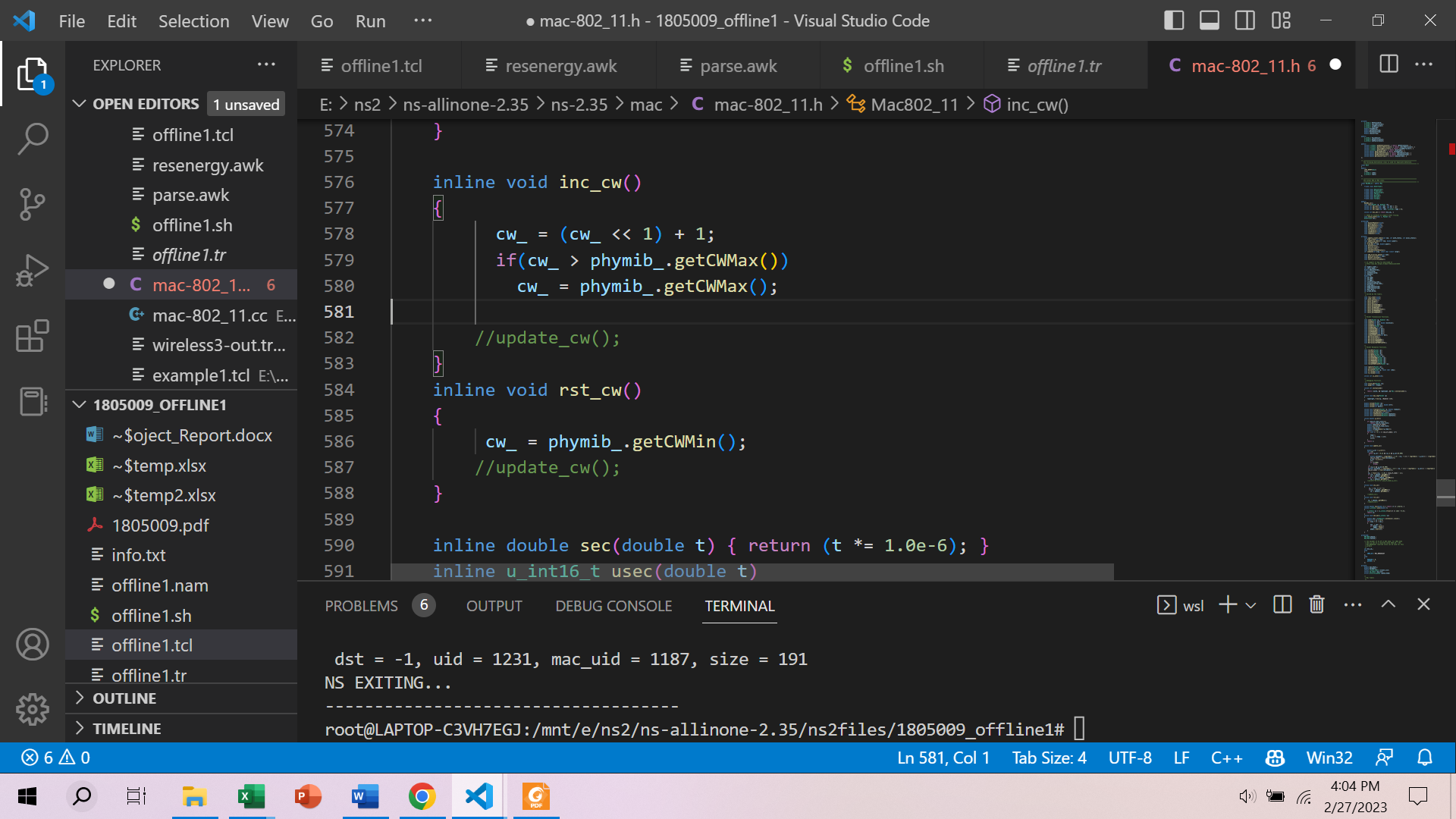
Email: [princezarzees5075@gmail.com](mailto:princezarzees5075@gmail.com)

**Supervisor**: NAVID BIN HASAN, Lecturer, Computer Science and Engineering, BUET

**Current implementation**

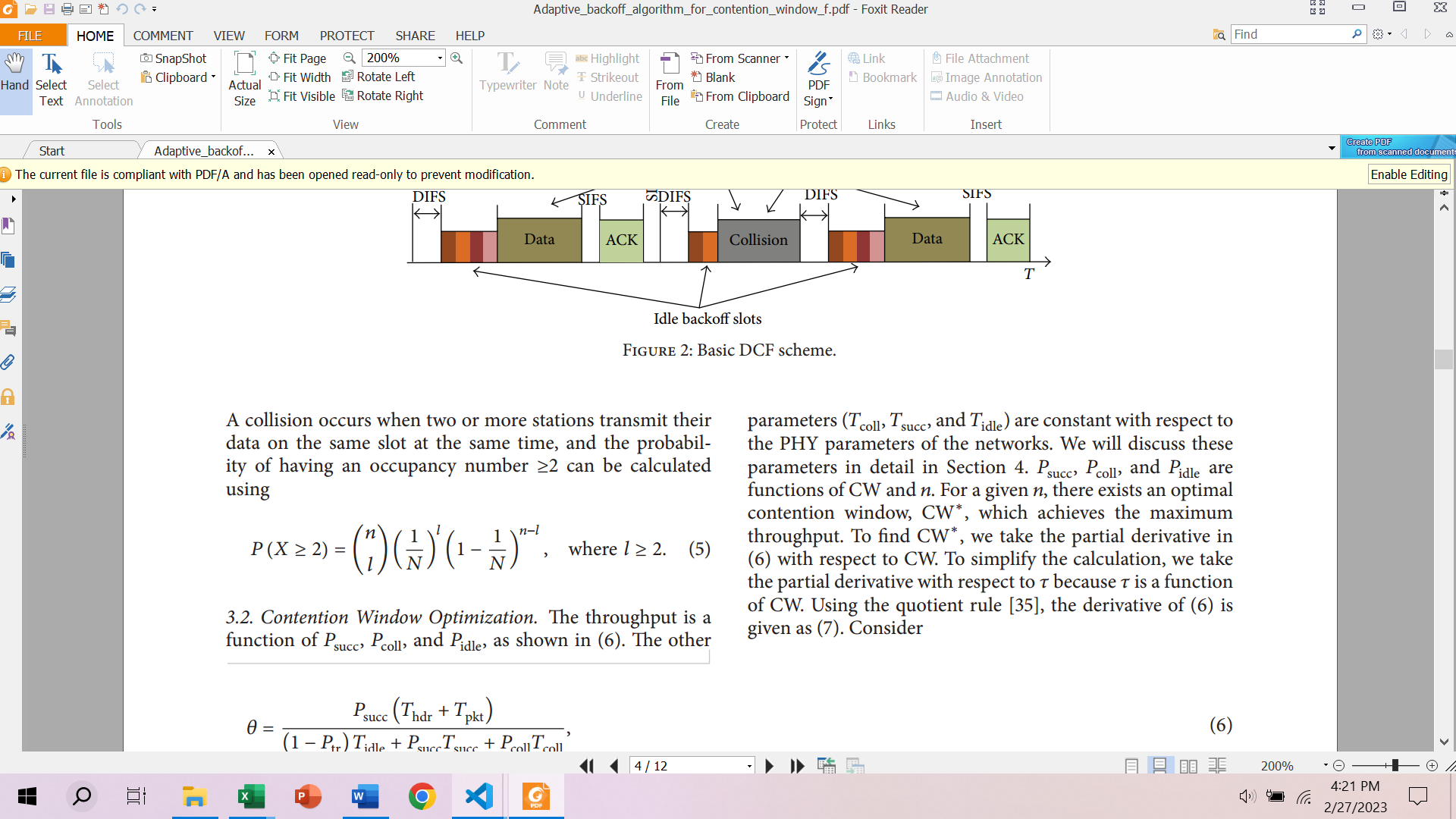
* Increasing contention window exponentially when detecting a collision ( DCF/Binary Exponential Backoff (BEB) )
* Resetting contention window after a successful transmission

**In mac-802\_11.h of default NS2**



**Modification made in NS2**

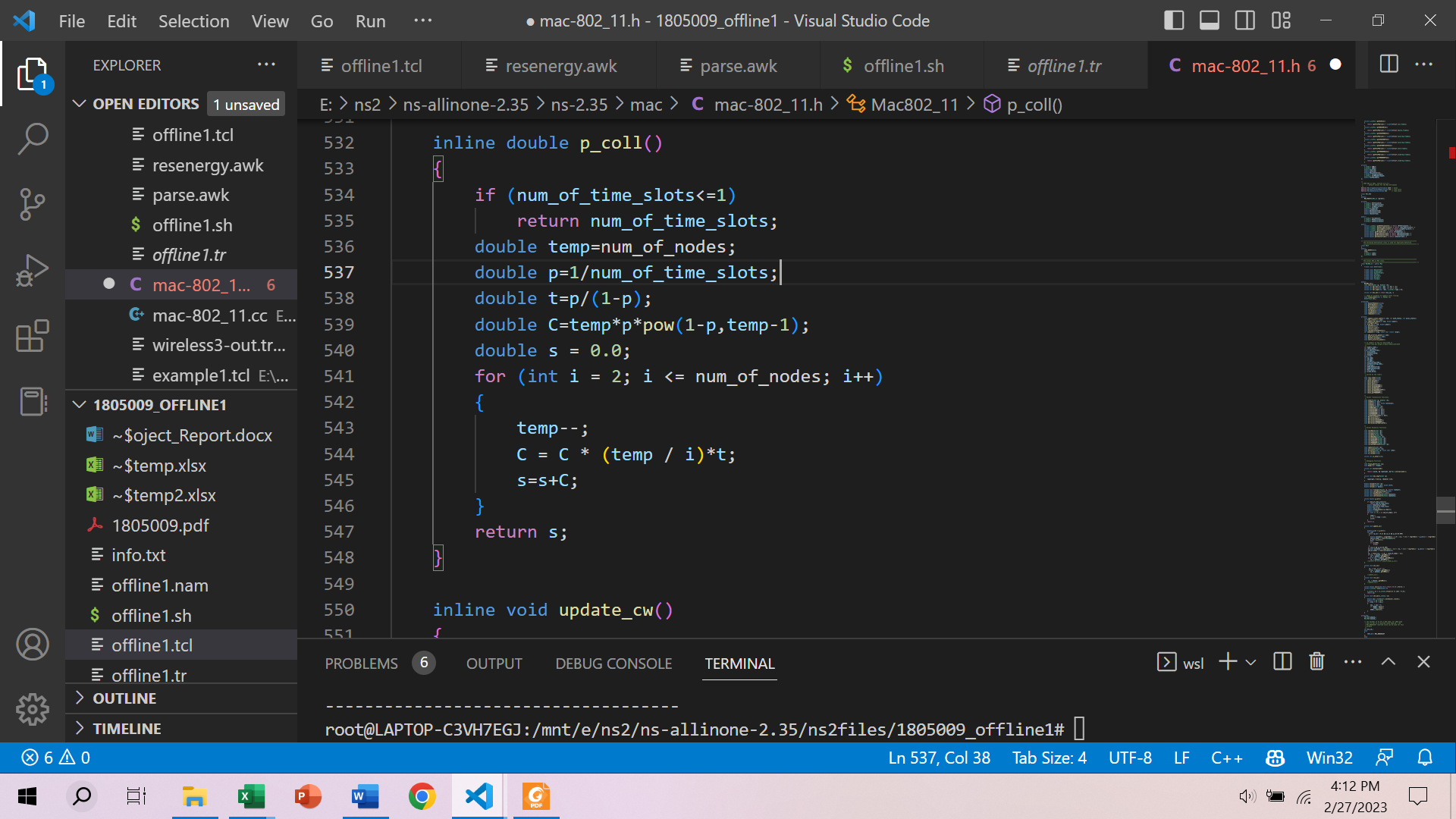
Calculating Probability

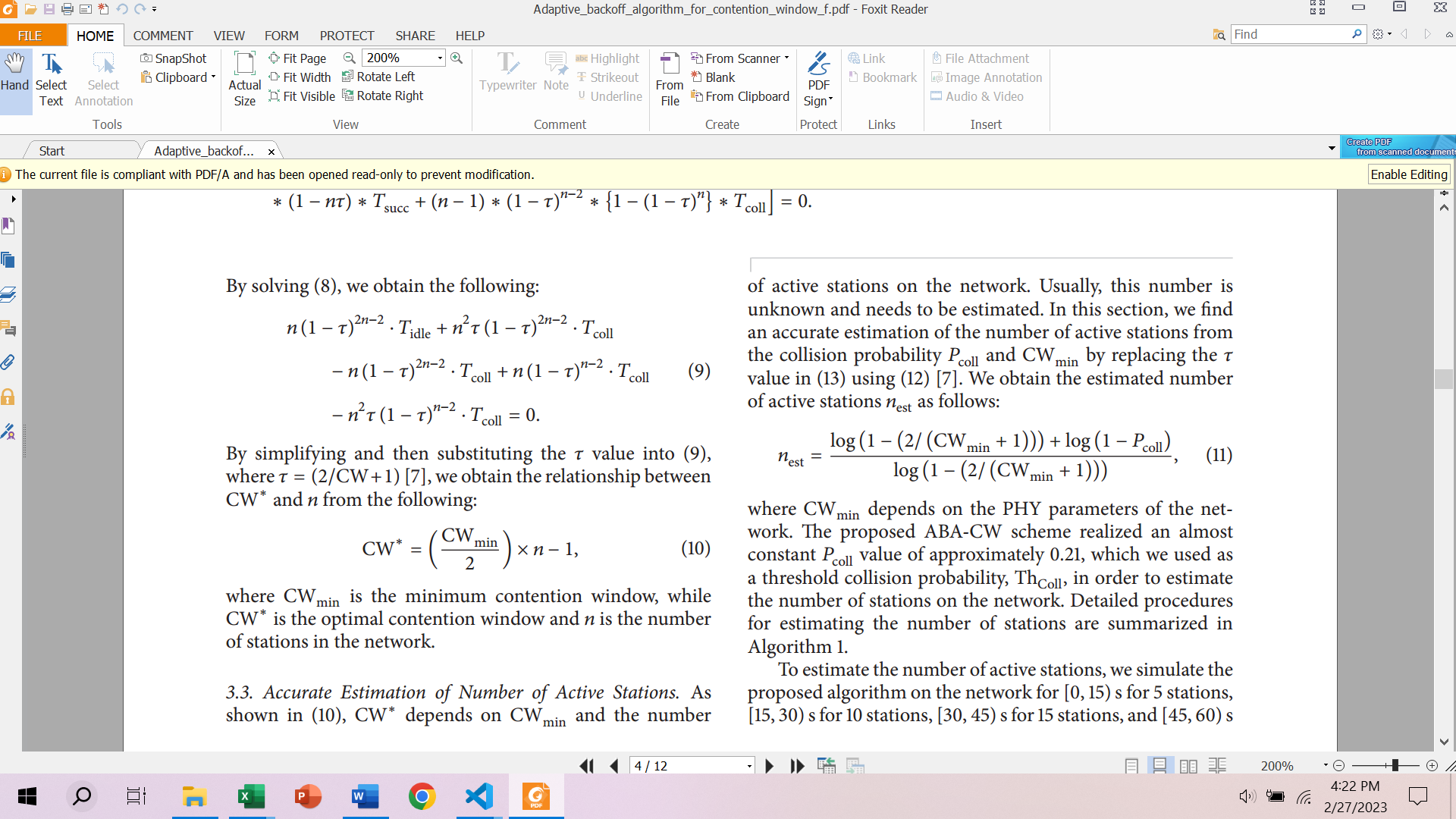


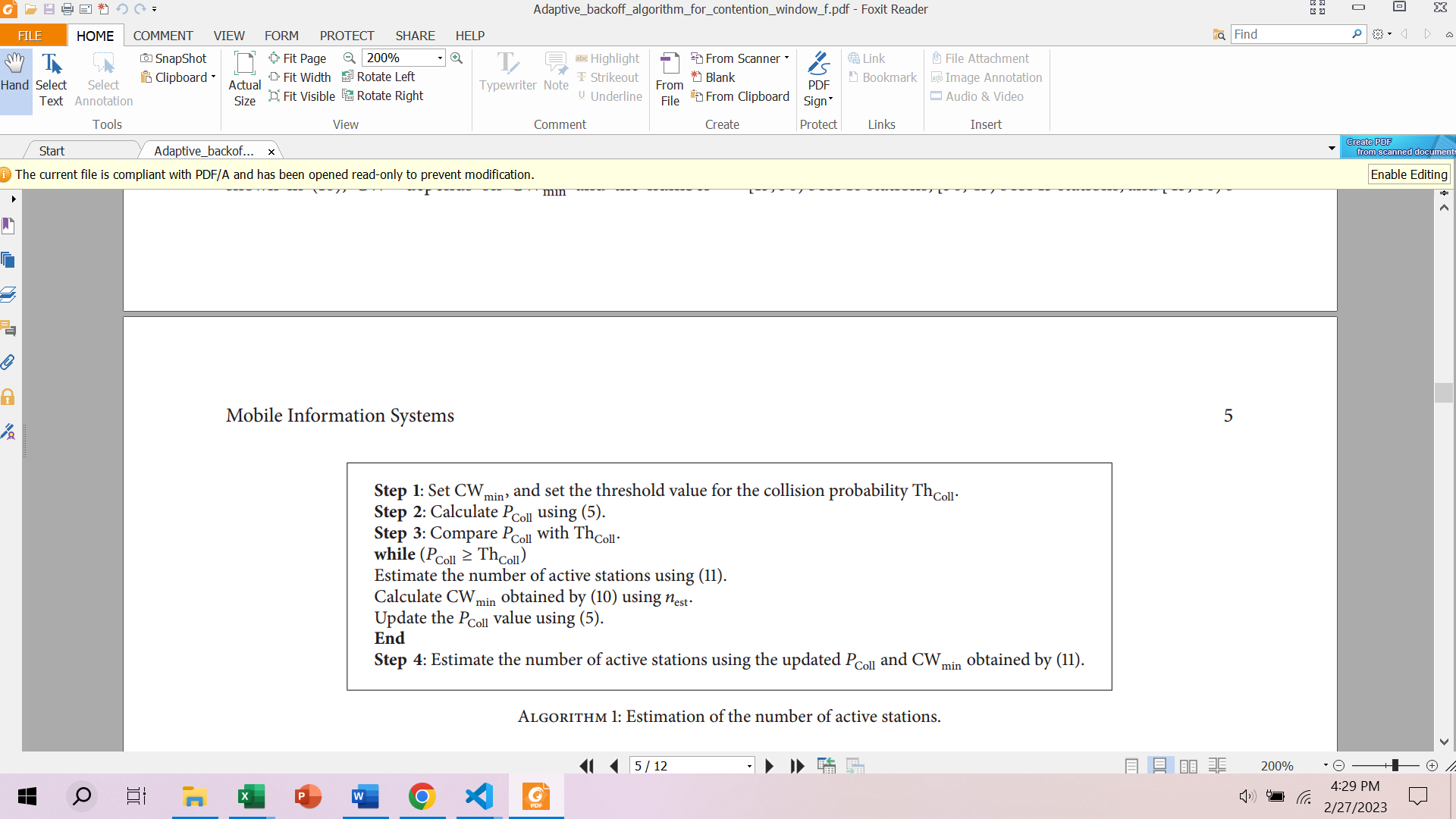
**Note:** In the article, they took as the contention number of a time slot. But in ns2 calculating contention number was not feasible. So I modified it in binomial distribution to calculate collision probability by summing over the probability of transmitting more than one node from = 2 to = n.

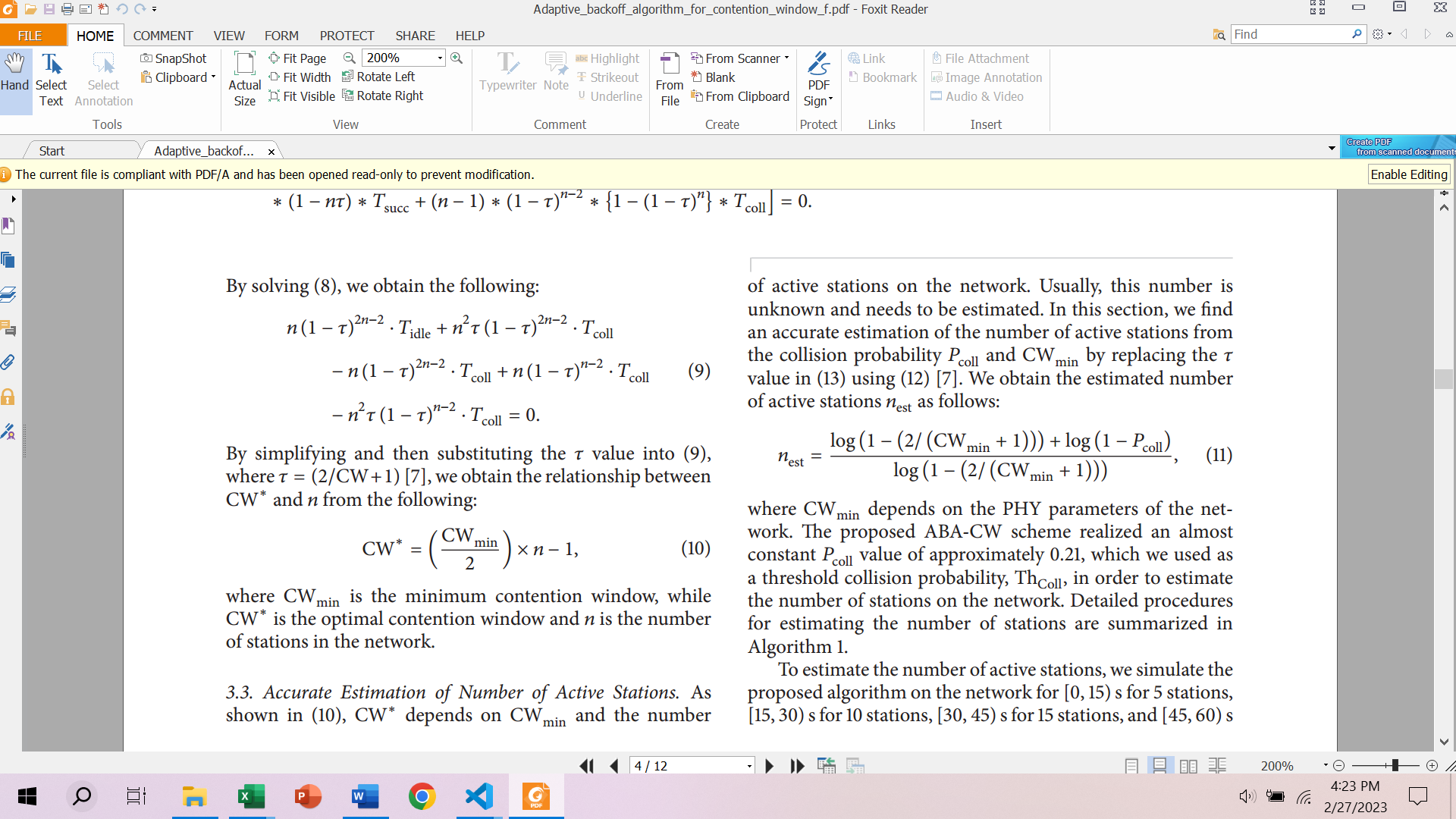
Here is the formula I used

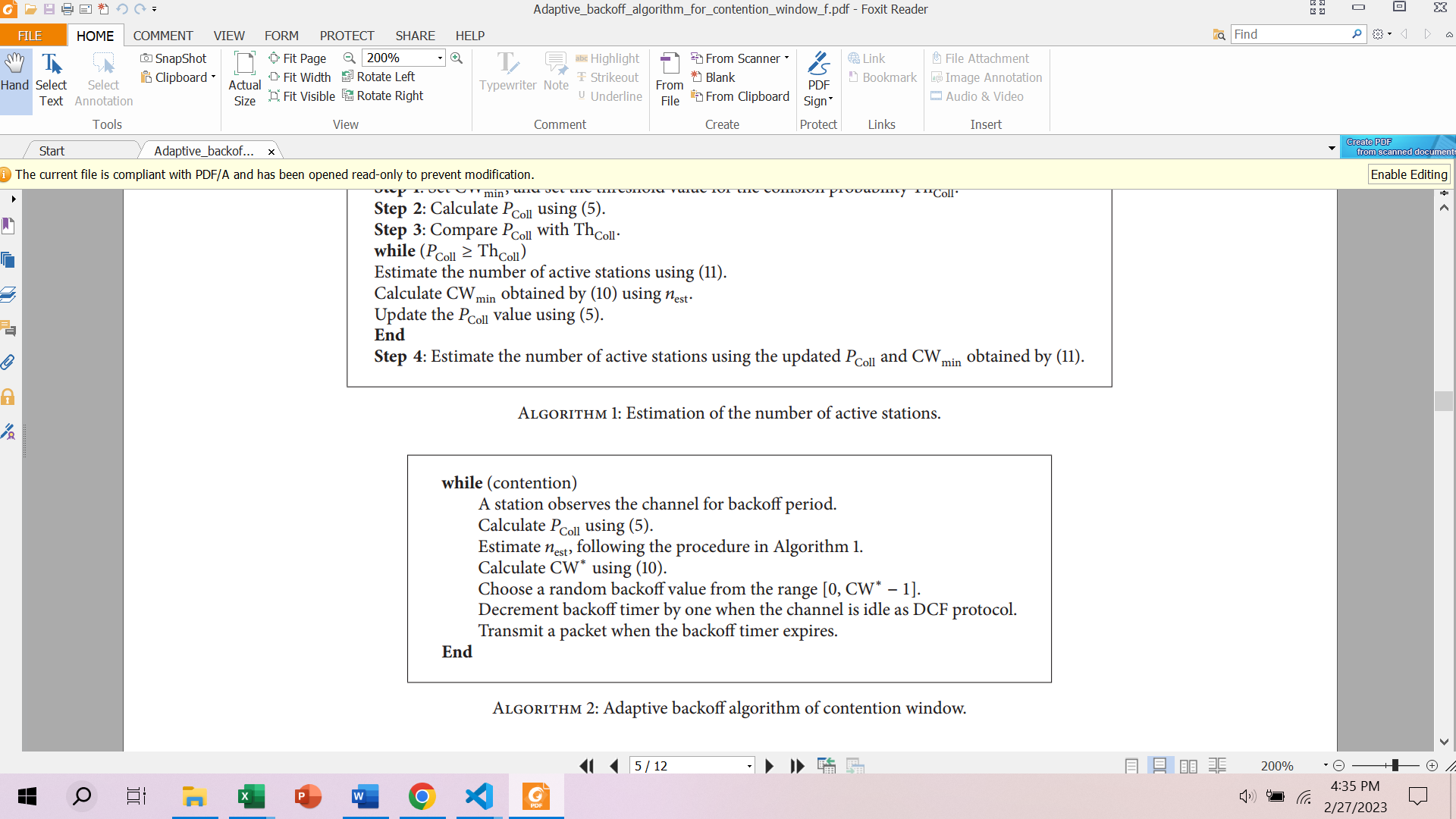
*Pcoll*



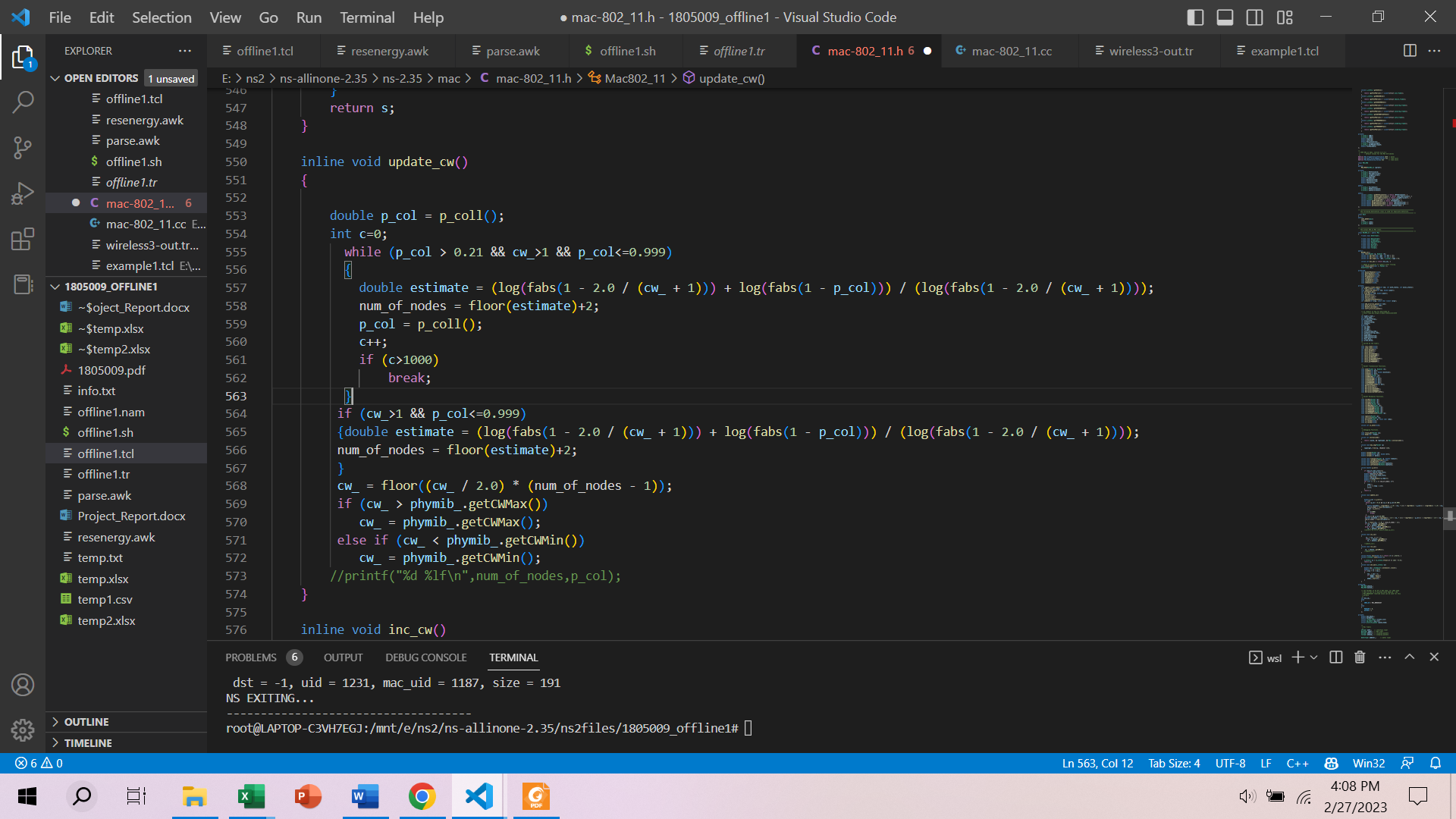




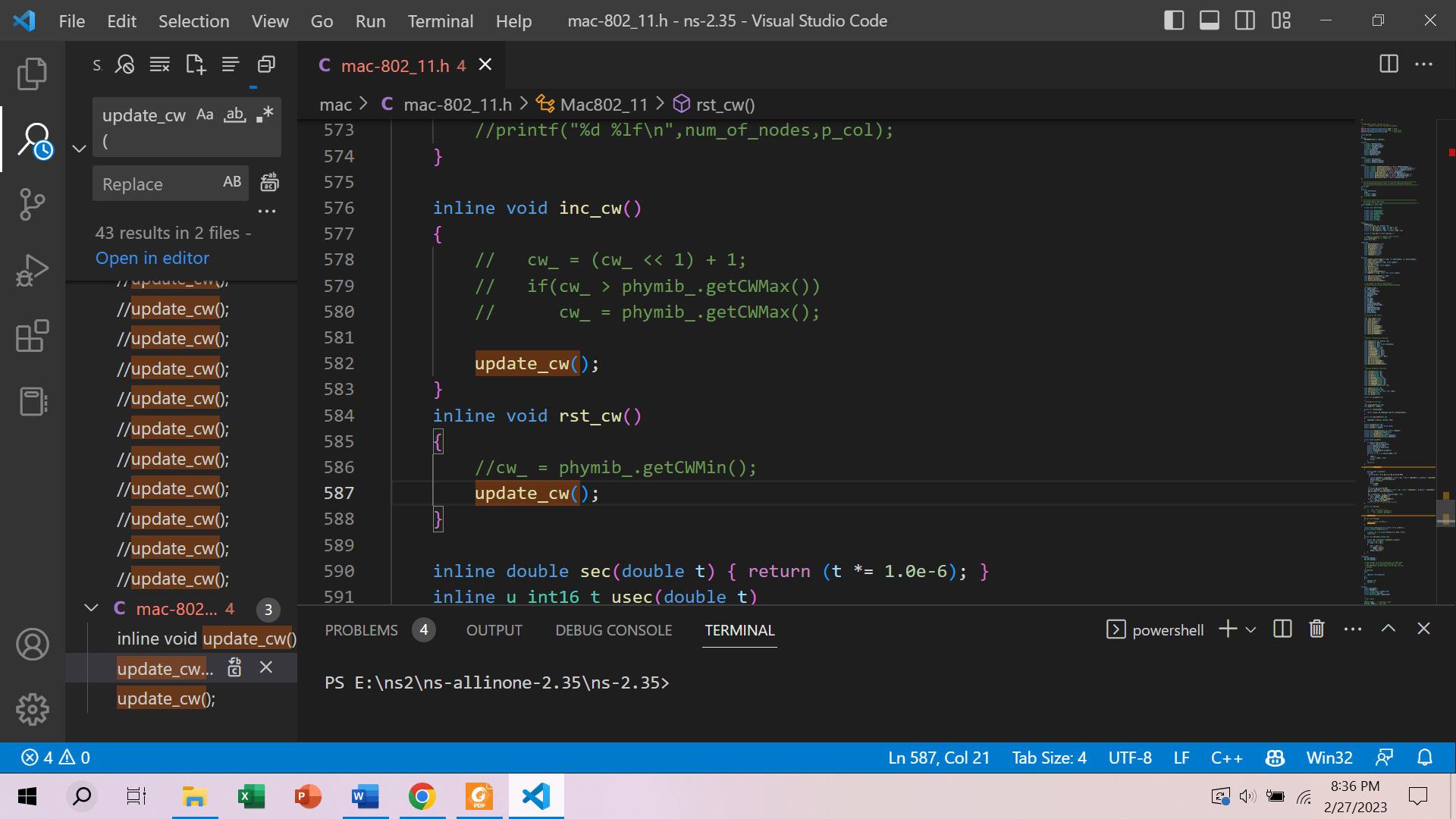




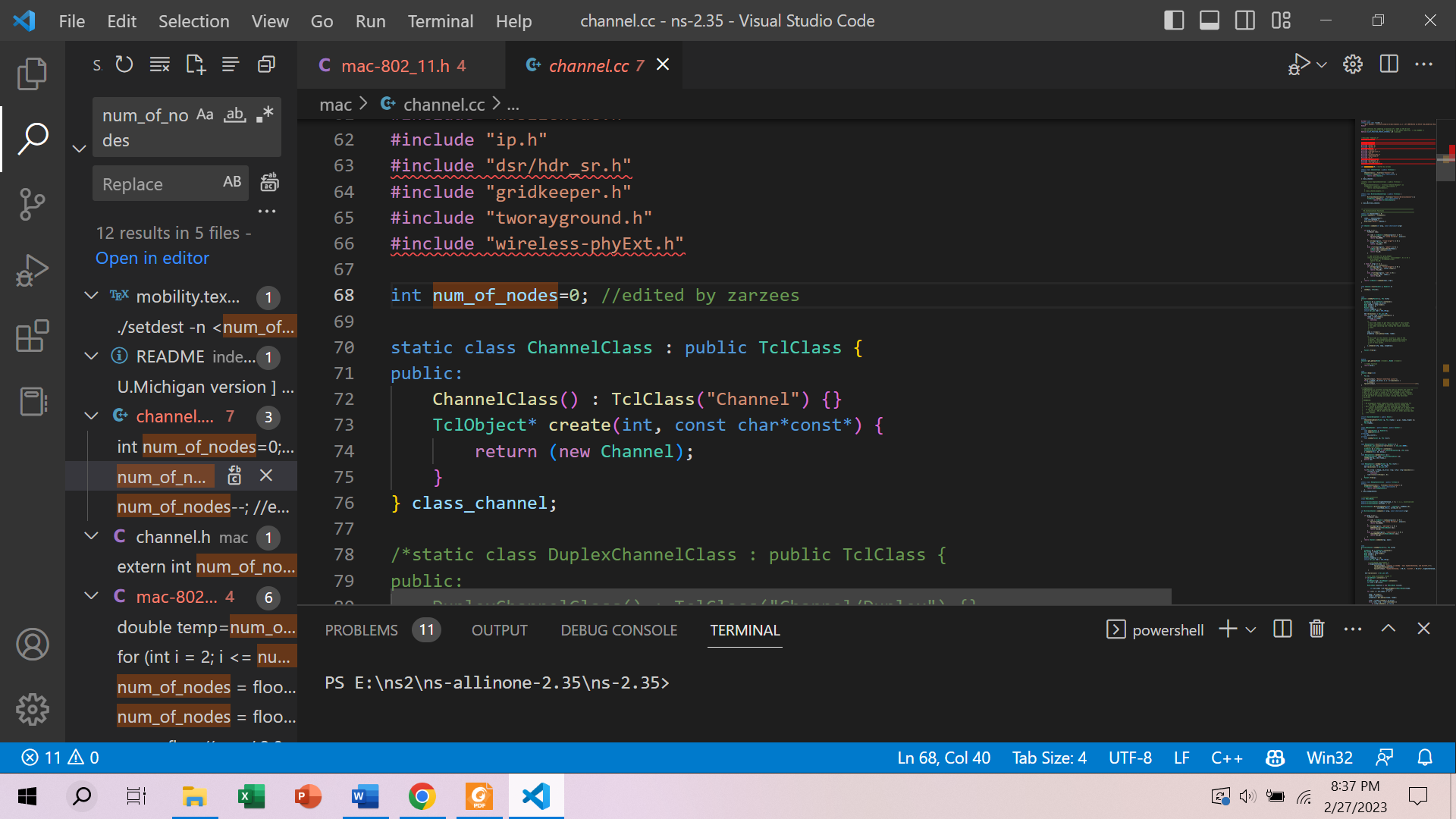
All the other formulas are as it is as in the research article. Here is the code for updating contention window.

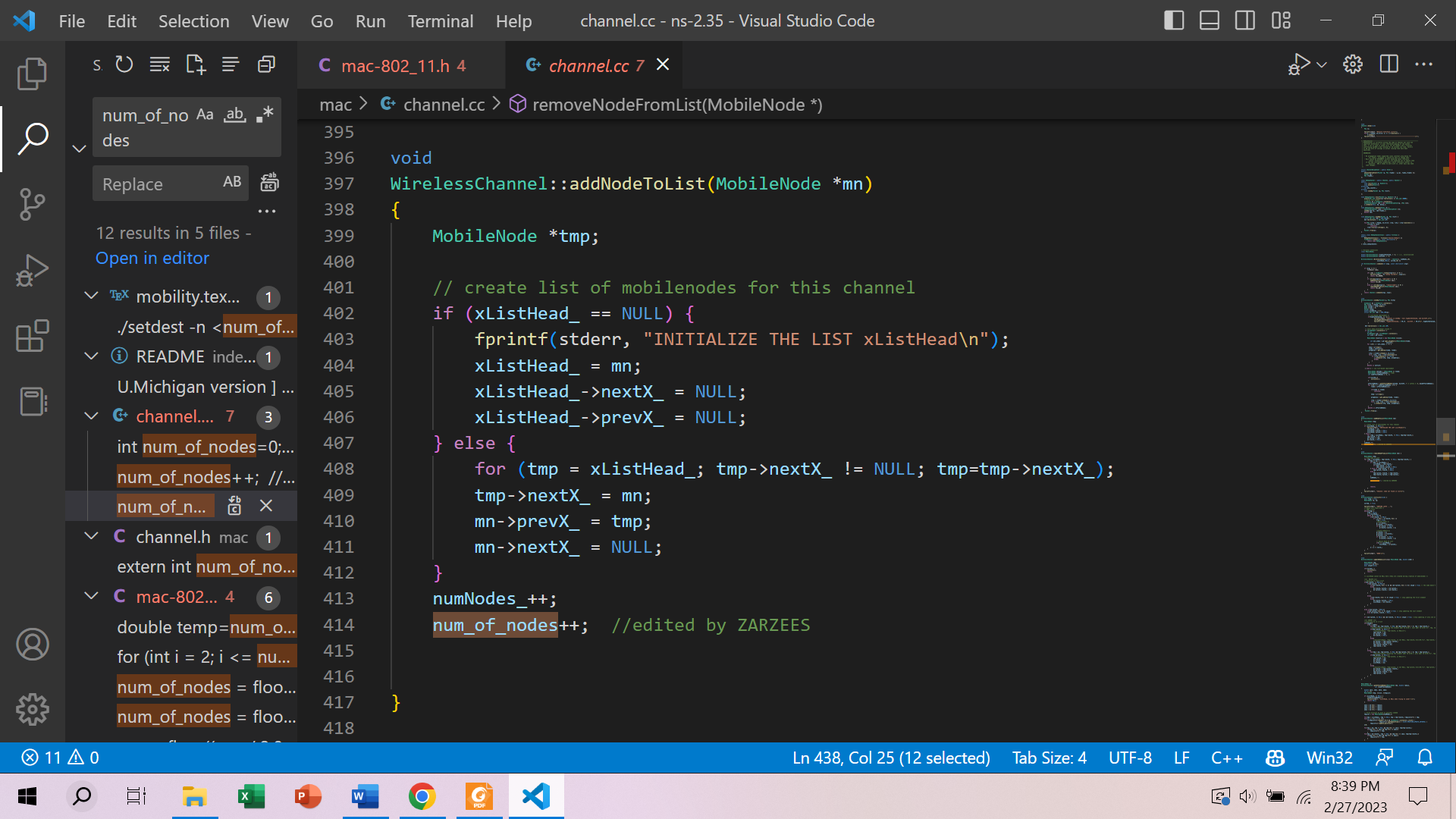


Wherever **inc\_cw()** and **rst\_cw()** is called, I replaced it with **update\_cw()**.

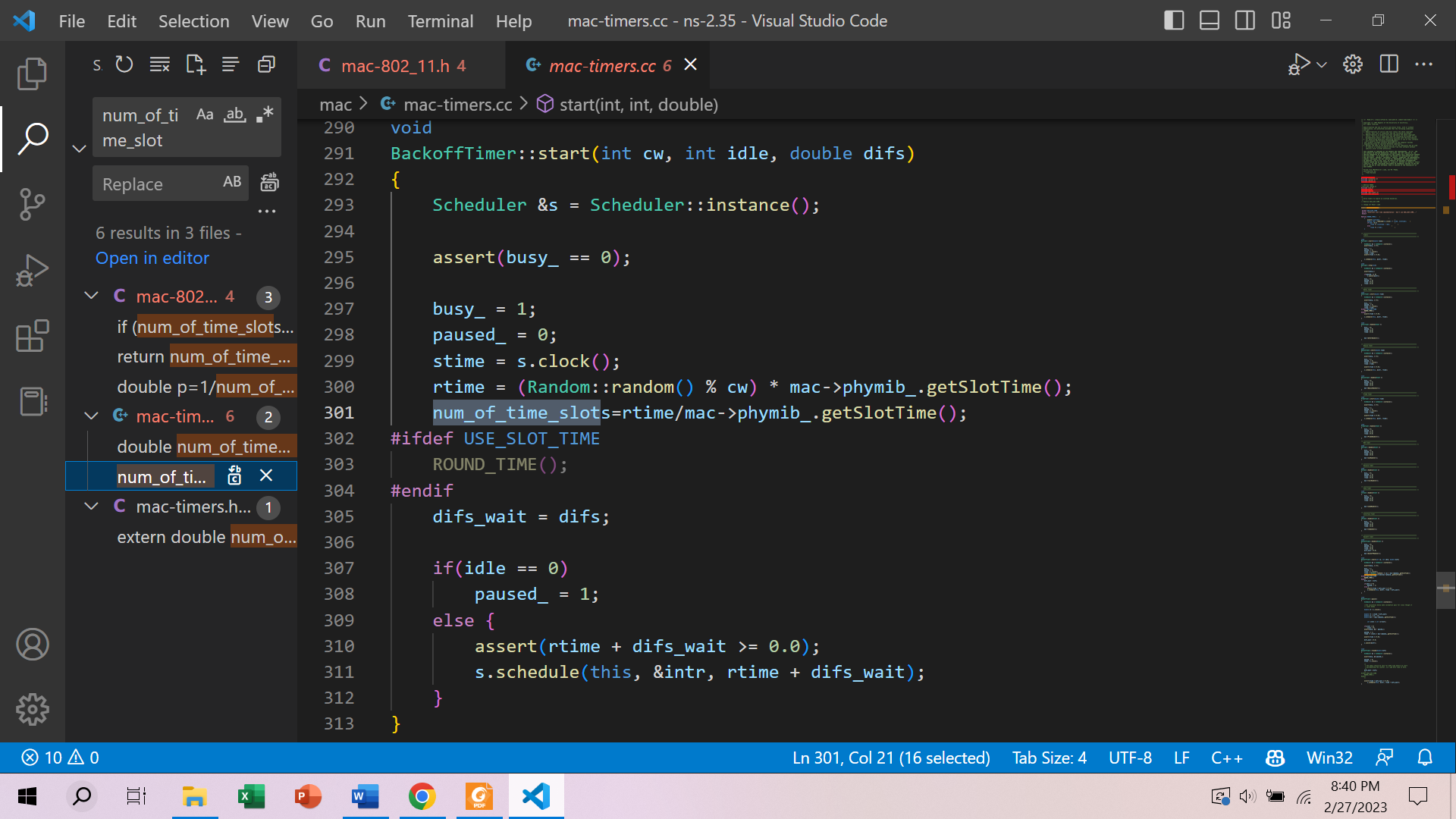


In channel.cc a global variable “**num\_of\_nodes**” is declared to get total number of nodes.





In mac-timers.cc a variable “**num\_of\_time\_slots**” is used to get time slots at that time.



**Network Topologies and Parameters Under Simulation:**

1. Wireless (802.11 and 802.15.4)
2. Mobility: Static
3. Positioning of Nodes (Grid 500×500)
4. Number of Nodes: 20, 40, 60, 80, 100
5. Number of Flows: 10, 20, 30, 40, 50
6. Packets per second: 100, 200, 300, 400, 500
7. Coverage: 1, 2, 3, 4, 5 (×TxRange, 1 for 100m)

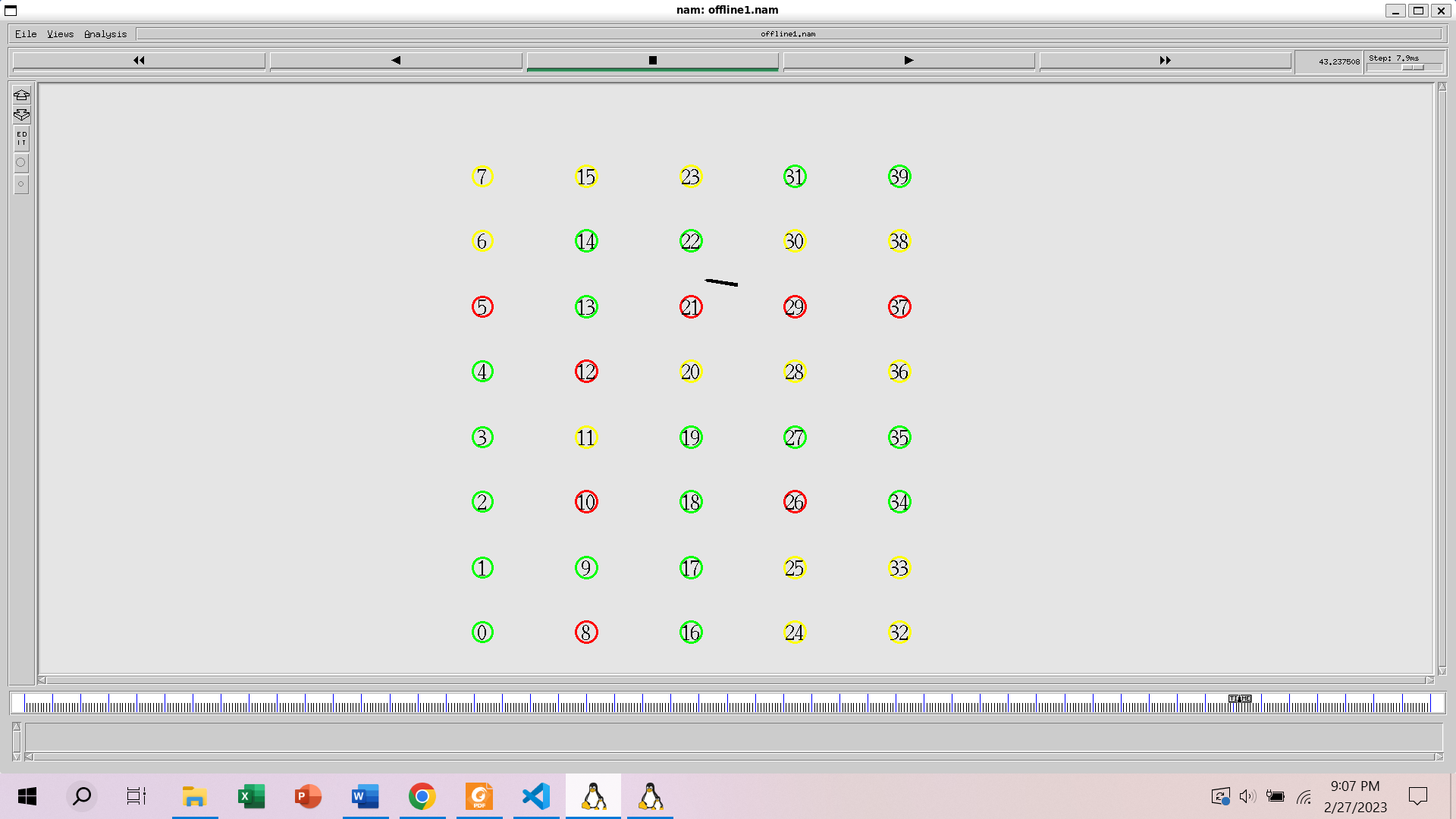
When varying parameters, other parameters are taken as follows

* Number of nodes 40
* Number of Flows 20
* Packets per second 200
* Coverage 3×TxRange

1. Simulation time 50 seconds

**Procedure:**

A bash script(project.sh) was run including all the necessary commands for all the parameters. Output was redirected to a csv(temp1.csv) file and all the graphs were generated by Excel. Parsing was done by parse.awk file and the simulation is in project.nam.



**Figure: A simulation snapshot of nam**

**Results:**

**Table 1 Varying Number of Nodes**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Number of Nodes** | **Throughput OLD** | **Average Delay OLD** | **Delivery ratio OLD** | **Drop ratio OLD** | **Energy Consumption OLD** | **Throughput NEW** | **Average Delay NEW** | **Delivery ratio NEW** | **Drop ratio NEW** | **Energy Consumption NEW** |
| 20 | 160936 | 0.0268599 | 0.998016 | 0.00198413 | 470.72 | 168990 | 0.073627 | 0.99777 | 0.00222965 | 469.51 |
| 40 | 173546 | 0.0673481 | 0.999077 | 0.000922509 | 952.376 | 167997 | 0.0618989 | 1 | 0 | 941.628 |
| 60 | 180563 | 0.107372 | 1 | 0 | 1414.75 | 219425 | 0.136935 | 0.996967 | 0.00303337 | 1402.9 |
| 80 | 198701 | 0.158408 | 0.999197 | 0.000803213 | 1899.8 | 186958 | 0.141907 | 0.997732 | 0.00199402 | 1879.77 |
| 100 | 205884 | 0.101772 | 0.99845 | 0.00155039 | 2379.2 | 216029 | 0.173628 | 0.999163 | 0.00083682 | 2438.36 |

**Figure 1.1: Throughput(bits/sec) vs Number of nodes**

Figure 1.2: Average Delay(sec) vs Number of nodes

Figure 1.3: Delivery Ratio vs Number of nodes

Figure 1.4: Drop Ratio vs Number of nodes

Figure 1.5: Energy Consumption(Joule) vs Number of nodes

**Table 2 Varying Number of Flows**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Number of Flows** | **Throughput OLD** | **Average Delay OLD** | **Delivery ratio OLD** | **Drop ratio OLD** | **Energy Consumption OLD** | **Throughput NEW** | **Average Delay NEW** | **Delivery ratio NEW** | **Drop ratio NEW** | **Energy Consumption NEW** |
| 10 | 85059.2 | 0.0196029 | 1 | 0 | 919.529 | 84896 | 0.0234881 | 1 | 0 | 917.788 |
| 20 | 195904 | 0.0477103 | 1 | 0 | 948.515 | 191504 | 0.0903508 | 1 | 0 | 939.893 |
| 30 | 302520 | 0.127417 | 0.995253 | 0.0021097 | 978.275 | 268289 | 0.17418 | 0.999395 | 0.000604961 | 962.52 |
| 40 | 390381 | 0.894675 | 0.976819 | 0.0039968 | 998.3 | 366580 | 0.991706 | 0.996114 | 0.00345423 | 994.543 |
| 50 | 392920 | 1.14036 | 0.948151 | 0.00724361 | 999.218 | 395194 | 1.43108 | 0.984974 | 0.00355872 | 995.695 |

Figure 2.1: Throughput vs Number of flows

Figure 2.2: Average Delay vs Number of flows

Figure 2.3: Delivery Ratio vs Number of flows

Figure 2.4: Drop Ratio vs Number of flows

Figure 2.5: Energy Consumption vs Number of flows

**Table 3 Varying Packets per second**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Packets Per Second** | **Throughput OLD** | **Average Delay OLD** | **Delivery ratio OLD** | **Drop ratio OLD** | **Energy Consumption OLD** | **Throughput NEW** | **Average Delay NEW** | **Delivery ratio NEW** | **Drop ratio NEW** | **Energy Consumption NEW** |
| 100 | 144822 | 0.0192375 | 1 | 0 | 926.943 | 175833 | 0.0729386 | 1 | 0 | 944.706 |
| 200 | 206189 | 0.0333827 | 0.999222 | 0.00077821 | 949.629 | 187782 | 0.0734348 | 0.999146 | 0.000853971 | 952.02 |
| 300 | 212554 | 0.0340293 | 0.997738 | 0 | 952.677 | 152664 | 0.0758007 | 0.997917 | 0 | 949.692 |
| 400 | 152170 | 0.0212006 | 1 | 0 | 933.875 | 199331 | 0.0574319 | 1 | 0 | 946.965 |
| 500 | 153146 | 0.0300373 | 0.998957 | 0.00104275 | 941.256 | 176650 | 0.0621937 | 0.999094 | 0.000905797 | 951.256 |

Figure 3.1: Throughput vs Number of Packets

Figure 3.2: Average Delay vs Number of packets

Figure 3.3: Delivery Ratio vs Number of packets

Figure 3.4: Drop Ratio vs Number of packets

Figure 3.5: Energy Consumption vs Number of packets

**Table 4 Varying TxRange**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TxRange** | **Throughput OLD** | **Average Delay OLD** | **Delivery ratio OLD** | **Drop ratio OLD** | **Energy Consumption OLD** | **Throughput NEW** | **Average Delay NEW** | **Delivery ratio NEW** | **Drop ratio NEW** | **Energy Consumption NEW** |
| 1 | 198907 | 0.403669 | 0.991987 | 0.00560897 | 981.407 | 160903 | 1.34559 | 0.95351 | 0.00759013 | 986.351 |
| 2 | 168659 | 0.0511213 | 0.998108 | 0.00189215 | 955.069 | 178447 | 0.204903 | 0.999103 | 0.000896861 | 956.003 |
| 3 | 188397 | 0.0275268 | 0.997451 | 0.00169924 | 944.103 | 223155 | 0.0599929 | 0.997245 | 0.00183655 | 946.504 |
| 4 | 168816 | 0.0151073 | 0.998108 | 0.000946074 | 931.167 | 175992 | 0.0487446 | 0.99909 | 0 | 936.673 |
| 5 | 197702 | 0.0131114 | 1 | 0 | 934.827 | 210592 | 0.0220871 | 0.998476 | 0 | 937.417 |

Figure 4.1: Throughput vs TxRange

Figure 4.2: Average Delay vs TxRange

Figure 4.3: Delivery Ratio vs TxRange

Figure 4.4: Drop Ratio vs TxRange

Figure 4.5: Energy Consumption vs TxRange

**Summary Findings**

According to my selected research article, throughput should be improved and collision rate should be reduced. But as I had to change the formula for probability of collision (finding contention number of a slot was not feasible in NS2) to binomial distribution, a vast improvement could not be found. Though Figure 1.1, 3.1, 4.1 shows some improvement over the built in method. Figure 2.1 gives almost same result.

Delay is more or less higher than the default, cause finding the perfect contention window through a while loop taking some time. See Figure 1.2, 2,2, 3.2 and 4.2.

Figure 1.3, 2.3 show some improvement in delivery ratio. 3.3 and 4.3 almost give same result as previous.

Drop ratio shows some clear improvement specially in Figure 2.4 and 4.4. Even it is clear from Figure 1.4 that when number of nodes increased, drop ratio decreased with new method.

Surprisingly even increasing the overhead of calculating optimum contention window iteratively, does not consume that much energy. Sometimes it consumes less (Figure 2.5). Overall the energy consumption is same.

In a nutshell, there was not that much colossal improvement in all cases. But throughput, delivery ratio and drop ratio give better result overall in majority cases. It might happen due to changing the formula of collision probability. As the writers show using the original formula, there will be a much better result.

Since using naïve probability distribution like binomial, giving better result in throughput and drop ratio, it can be said that the authors’ claim is correct about using adaptive back off algorithm for contention window.

**Additional Simulation (Not part of project)**

In 802.15.4