CS 342 Project 3 Report

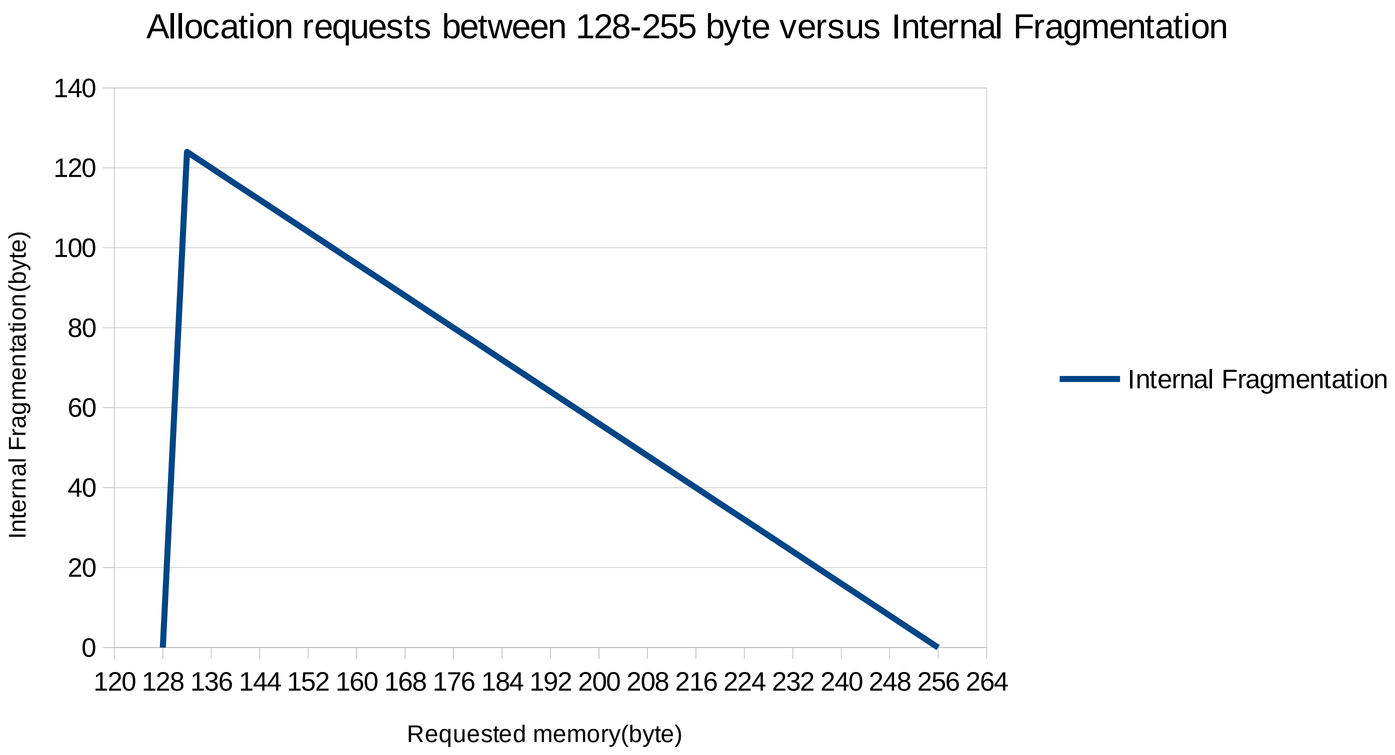
Muhammed Emre YILDIZ
21702825

Murat ANGIN
21702962

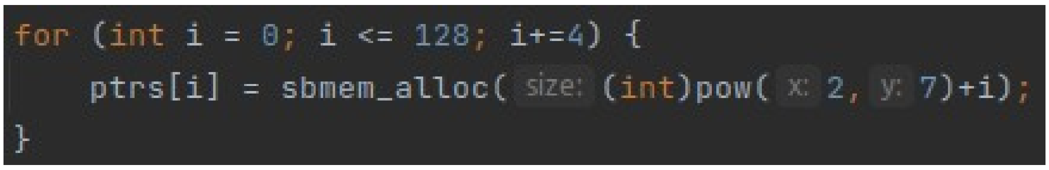
Introduction

We performed four different tests on our implementation of buddy algorithm. In order to detect and analyze the most powerful and weak points of buddy algorithm, we choose two different use case scenarios. As a comparison thirdly we made another test with random memory allocation request sizes. Lastly we developed a test case which shows us the effect of external fragmentation.

1. The Worst Case Scenario



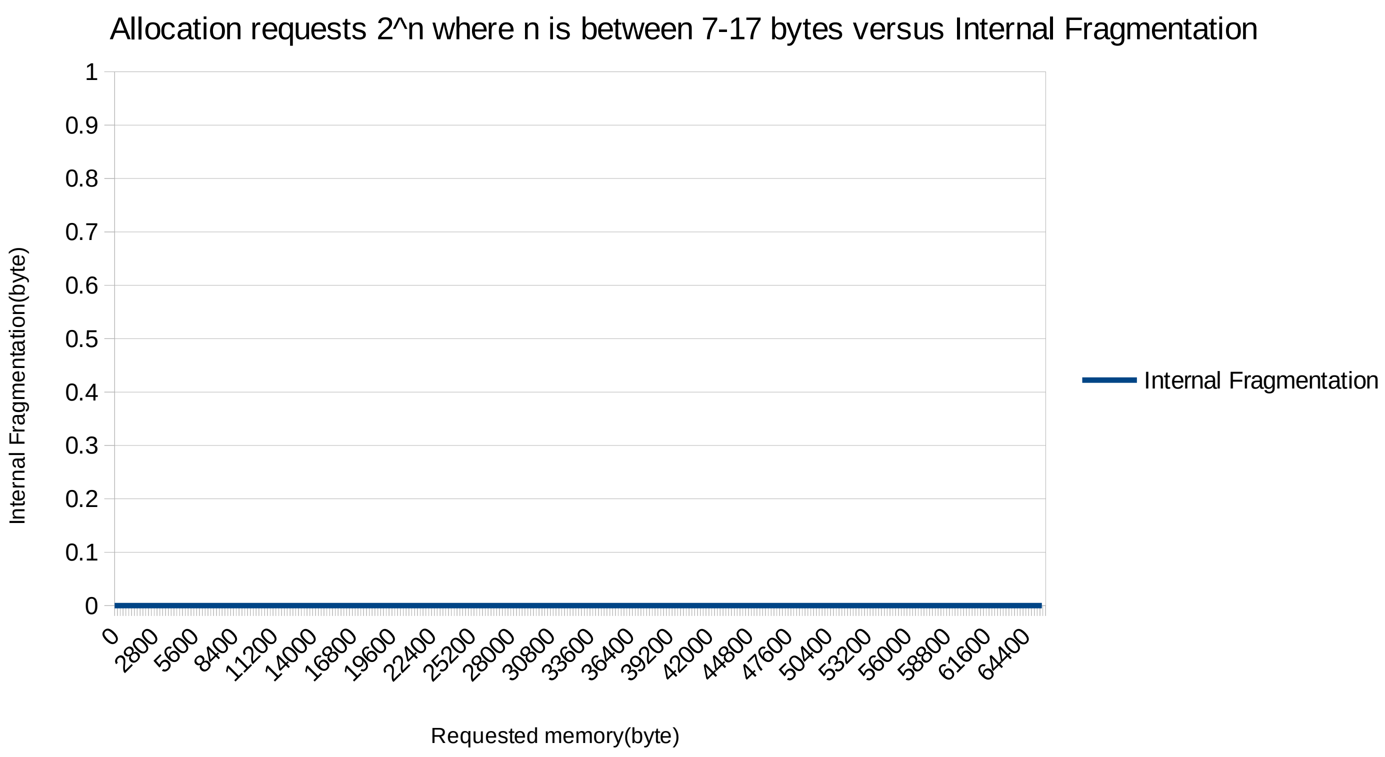
In this scenario we implemented a program which requests memory blocks starting from 128 bytes to 256 bytes with step size equals to 4.



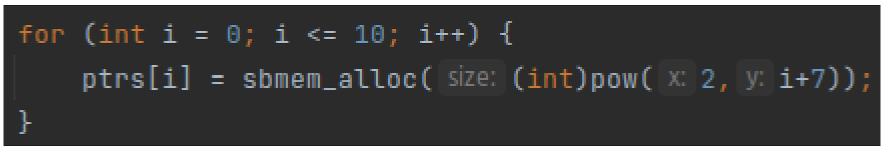
```
for (int i = 0; i <= 128; i+=4) {
    ptrs[i] = sbmem_alloc( size: (int)pow( x: 2, y: 7)+i);
}
```

As a result, we observed that when the requested amount of memory becomes closer to the power of 2, the internal fragmentation amount decreases linearly. When the requested memory size was equals to power of 2, the internal fragmentation amount equals to 0 and becomes maximum after this value. This behavior shows that the reason of the pike occurs after 128 byte of allocation.

2. The Best Case Scenario



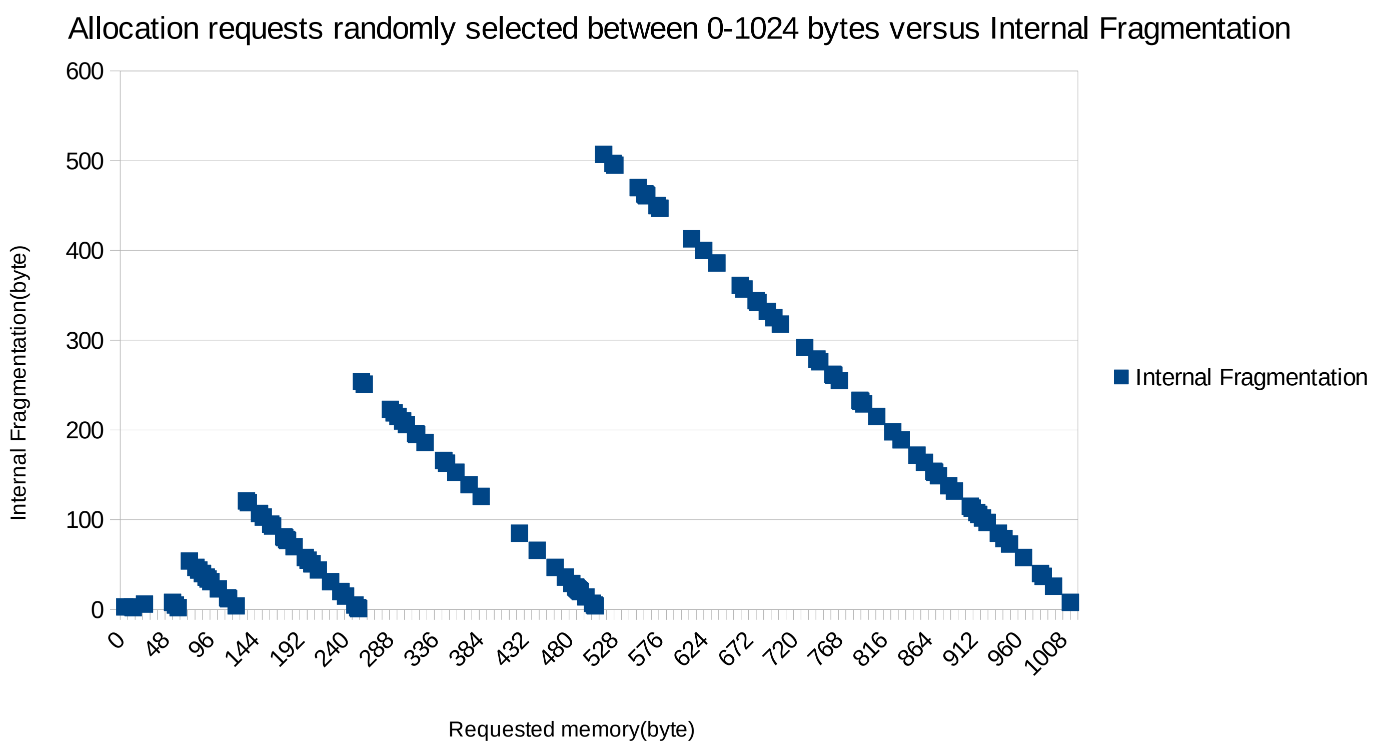
In this scenario we implemented a program which requests memory blocks starting from 128 bytes to 2^{17} bytes exponentially.



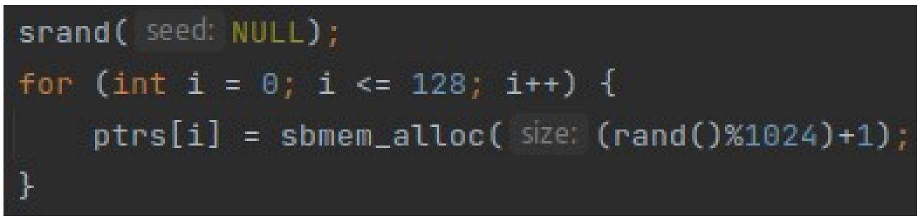
```
for (int i = 0; i <= 10; i++) {
    ptrs[i] = sbmem_alloc( size: (int)pow( x: 2, y: i+7));
}
```

As a result, as we observed already in the worst case scenario, when the requested amount of memory equals to power of 2, the internal fragmentation amount will be 0. The main cause of this result is the division strategy of buddy algorithm. In the implementation of buddy algorithm we divided memory into chunks which consists of power of 2 bytes. Therefore every allocation equals to power of 2 fitted perfectly to the allocated chunk.

3. Random Case Scenario



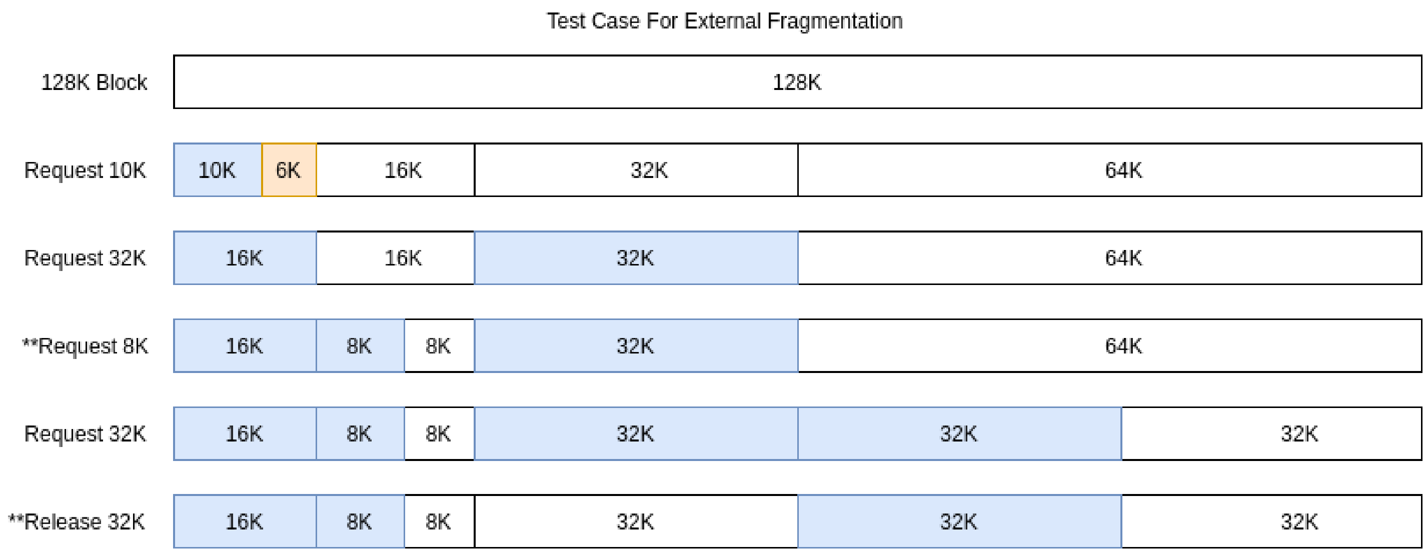
In this scenario we implemented a program which requests memory blocks starting from 1 bytes to 1024 bytes randomly.



```
srand( seed: NULL);
for (int i = 0; i <= 128; i++) {
    ptrs[i] = sbmem_alloc( size: (rand()%1024)+1);
}
```

As a result, as we observed already in the worst case scenario, when the requested amount of memory become closer to power of 2, the internal fragmentation amount will go to 0. Therefore the values such that 16,64,128 and 512 has got zero internal fragmentation.

4. External Fragmentation Use Case Scenario



We draw a diagram in order to show different stages of the test scenario. Library initialized with a memory size equals to 128K. The first memory allocation request is 10K, therefore a chunk of 16K returned to the process because of the buddy allocation algorithm division. Therefore there is a internal fragmentation with the size 6K occurred. After this request, a couple of additional allocations requested. At the last stage of the test case, the free space library has divided into two different areas. Therefore, when a user of the library make a request with the size of 64K, we couldn’t give any memory chunk to the process even though there is 72K of free space exists in total. It is a weak point of the buddy algorithm. Because the 32k chunks currently free but they are not buddy of each other, they cannot be merged into a upper order chunk.
Thus, external fragmentation occurs in size of:

$$1-\frac{Largest\ Block\ of\ Free\ Memory}{Total\ Free\ Memory}=1-\frac{32\,K}{72\,K}=0.56$$

Conclusion

We observed that because of the nature of the buddy algorithm, the requests closer to power of 2 in terms of size are allocated in a more efficient way. After the free requests, the algorithm tries to find a buddy and merge them into a bigger chunk in high order in order to reduce external fragmentation. However, in some cases such as we investigate in the last test case, external fragmentation can be occurred respect to the order of requests.