CS342 Operating Systems
Spring 2021

Project 3
Memory Management

Musa Ege Ünalan 21803617

Mustafa Göktan Güdükbay 21801740

1. Experiment

We measured the internal fragmentation using two approaches. The first approach measured the ratio of average allocated space to average requested space and the second approach was the ratio of over-allocated space to memory size.

We first used exponential distribution to determine the size of requested spaces. We tried six average values (in bytes): 128, 256, 512, 1024, 2048, 4096 for four different segment sizes (in kilobytes): 32, 64, 128, 256. We allocated memory until it could not allocate any more.

Then we experimented on fixed sizes. We also used two methods in the fixed-size approach. We first took fixed sizes of (in bytes) 120, 248, 504, 1016, 2040, and 4088 to add up with overhead sizes to fill a block. We then tried fixed sizes of (in bytes) 128, 256, 512, 1024, 2048, 4096.

We measured external fragmentation by analyzing the ratio of the free memory to the total memory when a requested allocation cannot be made because no block can handle that request, although the total free memory is enough to handle the request.

2. Results

Tables 1, 2, 3, and 4 shows the results of fragmentation for segment sizes of 32 KB, 64 KB, 128 KB, and 256 KB using exponential distributed sizes.

Tables 5 and 6 show Fragmentation using fixed segment size of 32 KB where the overhead is not taken into account and is taken into account, respectively.

Tables 7 and 8 show Fragmentation using fixed segment size of 64 KB where the overhead is not taken into account and is taken into account, respectively.

Tables 9 and 10 show Fragmentation using fixed segment size of 128 KB where the overhead is not taken into account and is taken into account, respectively.

Tables 11 and 12 show Fragmentation using fixed segment size of 256 KB where the overhead is not taken into account and is taken into account, respectively.

Figures 1, 2, 3, and 4 show internal fragmentation for the segment sizes of 32 KB, 64 KB, 128 KB, and 256 KB, respectively, using exponential distributed sizes.

Figures 5, 6, 7, and 8 show external fragmentation for the segment size of 32 KB, 64 KB, 128 KB, and 256 KB, respectively, using exponential distributed sizes.

Figures 9 and 10 show internal fragmentation for the fixed segment size of 32 KB where the overhead not taken into account and taken into account, respectively.

Figures 11 and 12 show external fragmentation for the fixed segment size of 32 KB where the overhead not taken into account and taken into account, respectively.

Requested Average Size	Ratio of Average Allocated Space to Average Requested Space	Ratio of Overallocated Space to Memory Size	Ratio of Unallocated Space to Memory Size
128	1.53	0.34	0.016
256	1.42	0.29	0.008
512	1.52	0.34	0.008
1024	1.45	0.31	0.016
2048	1.35	0.23	0.125
4096	1.41	0.26	0.094

Table 1 - Fragmentation (Segment Size: 32KB) Using Exponential Distributed Sizes

Requested Average Size	Ratio of Average Allocated Space to Average Requested Space	Ratio of Overallocated Space to Memory Size	Ratio of Unallocated Space to Memory Size
128	1.47	0.32	0.012
256	1.48	0.32	0.004
512	1.43	0.3	0.004
1024	1.40	0.28	0.027
2048	1.41	0.29	0.008
4096	1.44	0.29	0.062

Table 2 - Fragmentation (Segment Size 64KB) Using Exponential Distributed Sizes

Requested Average Size	Ratio of Average Allocated Space to Average Requested Space	Ratio of Overallocated Space to Memory Size	Ratio of Unallocated Space to Memory Size
128	1.51	0.34	0.002
256	1.48	0.32	0.006
512	1.5	0.33	0.002
1024	1.45	0.31	0.002
2048	1.44	0.3	0.002
4096	1.33	0.24	0.016

Table 3 - Fragmentation (Segment Size 128 KB) Using Exponential Distributed Sizes

Requested Average Size	Ratio of Average Allocated Space to Average Requested Space	Ratio of Overallocated Space to Memory Size	Ratio of Unallocated Space to Memory Size
128	1.5	0.34	0.001
256	1.48	0.33	0.002
512	1.42	0.29	0.006
1024	1.43	0.30	0.001
2048	1.38	0.27	0.002
4096	1.41	0.29	0.005

Table 4 - Fragmentation (Segment Size 256 KB) Using Exponential Distributed Sizes

Requested Size	Ratio of Average Allocated Space to Average Requested Space	Ratio of Overallocated Space to Memory Size	Ratio of Unallocated Space to Memory Size
128	2	0.50	0.008
256	2	0.49	0.016
512	2	0.48	0.031
1024	2	0.47	0.062
2048	2	0.44	0.125
4096	2	0.38	0.250

Table 5 - Fragmentation (Segment Size 32 KB) Using Fixed Sizes (Overhead is not taken into account)

Requested Size	Ratio of Average Allocated Space to Average Requested Space	Ratio of Overallocated Space to Memory Size	Ratio of Unallocated Space to Memory Size
120	1	0	0.008
248	1	0	0.008
504	1	0	0.016
1016	1	0	0.031
2040	1	0	0.062
4088	1	0	0.125

Table 6 - Fragmentation (Segment Size 32 KB) Using Fixed Sizes (Overhead is taken into account)

Requested Size	Ratio of Average Allocated Space to Average Requested Space	Ratio of Overallocated Space to Memory Size	Ratio of Unallocated Space to Memory Size
128	2	0.5	0.004
256	2	0.5	0.008
512	2	0.49	0.016
1024	2	0.48	0.031
2048	2	0.47	0.062
4096	2	0.44	0.125

Table 7 - Fragmentation (Segment Size 64 KB) Using Fixed Sizes (Overhead is not taken into account)

Requested Size	Ratio of Average Allocated Space to Average Requested Space	Ratio of Overallocated Space to Memory Size	Ratio of Unallocated Space to Memory Size
120	1	0	0.004
248	1	0	0.004
504	1	0	0.008
1016	1	0	0.016
2040	1	0	0.031
4088	1	0	0.062

Table 8 - Fragmentation (Segment Size 64 KB) Using Fixed Sizes (Overhead is taken into account)

Requested Size	Ratio of Average Allocated Space to Average Requested Space	Ratio of Overallocated Space to Memory Size	Ratio of Unallocated Space to Memory Size
128	2	0.5	0.002
256	2	0.5	0.004
512	2	0.5	0.008
1024	2	0.49	0.016
2048	2	0.48	0.031
4096	2	0.47	0.062

Table 9 - Fragmentation (Segment Size 128 KB) Using Fixed Sizes (Overhead is not taken into account)

Requested Size	Ratio of Average Allocated Space to Average Requested Space	Ratio of Overallocated Space to Memory Size	Ratio of Unallocated Space to Memory Size
120	1	0	0.002
248	1	0	0.002
504	1	0	0.004
1016	1	0	0.008
2040	1	0	0.016
4088	1	0	0.031

Table 10 - Fragmentation (Segment Size 128 KB) Using Fixed Sizes (Overhead is taken into account)

Requested Size	Ratio of Average Allocated Space to Average Requested Space	Ratio of Overallocated Space to Memory Size	Ratio of Unallocated Space to Memory Size
128	2	0.5	0.001
256	2	0.5	0.002
512	2	0.5	0.004
1024	2	0.5	0.008
2048	2	0.49	0.016
4096	2	0.48	0.031

Table 11 - Fragmentation (Segment Size 256 KB) Using Fixed Sizes (Overhead is not taken into account)

Requested Size	Ratio of Average Allocated Space to Average Requested Space	Ratio of Overallocated Space to Memory Size	Ratio of Unallocated Space to Memory Size
120	1	0	0.001
248	1	0	0.001
504	1	0	0.002
1016	1	0	0.004
2040	1	0	0.008
4088	1	0	0.016

Table 12 - Fragmentation (Segment Size 256 KB) Using Fixed Sizes (Overhead is taken into account)

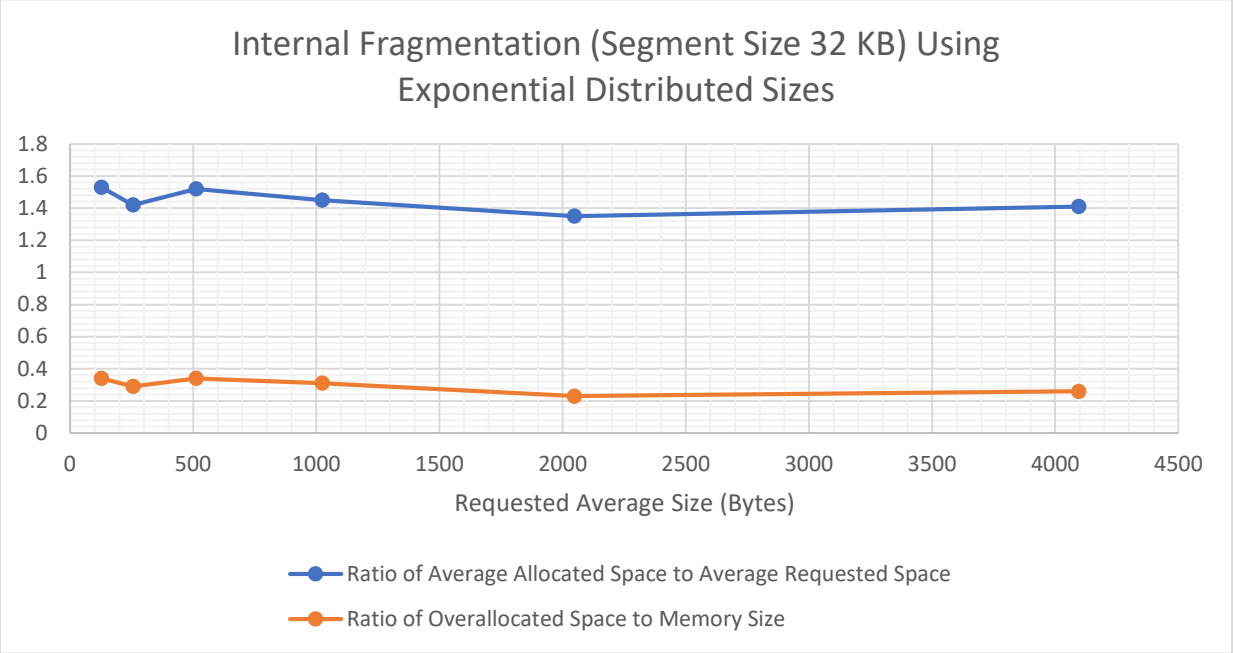

Figure 1. Internal Fragmentation (Segment Size 32 KB) Using Exponential Distributed Sizes

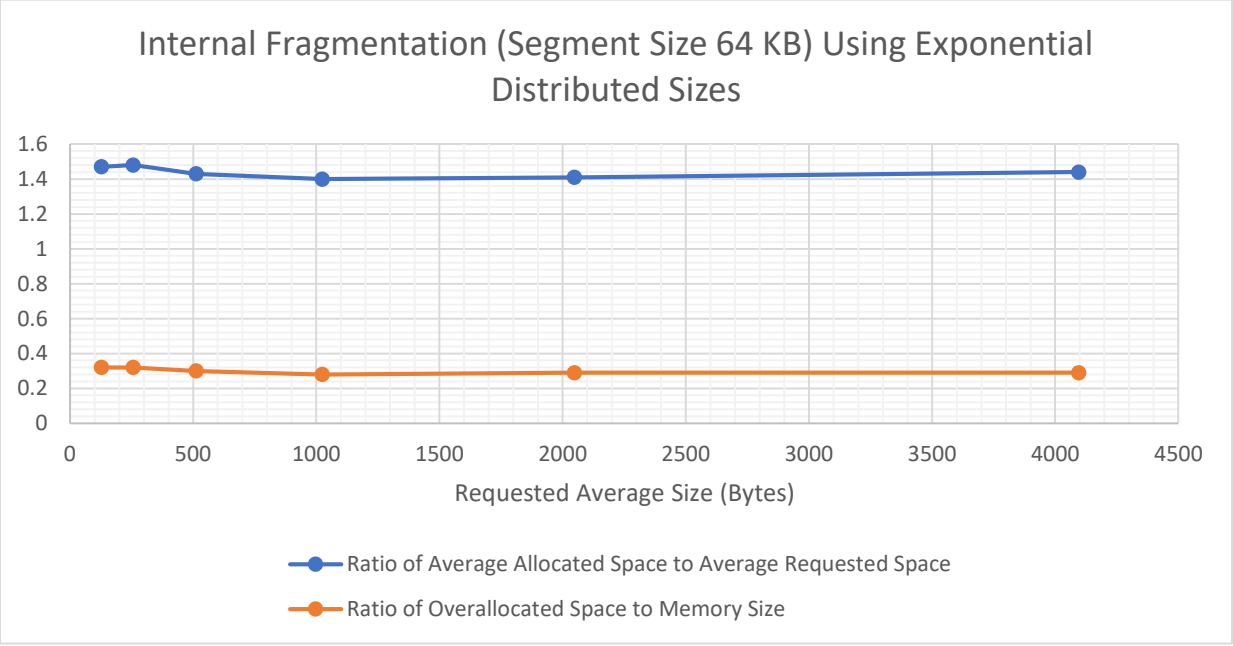

Figure 2. Internal Fragmentation (Segment Size 64 KB) Using Exponential Distributed Sizes

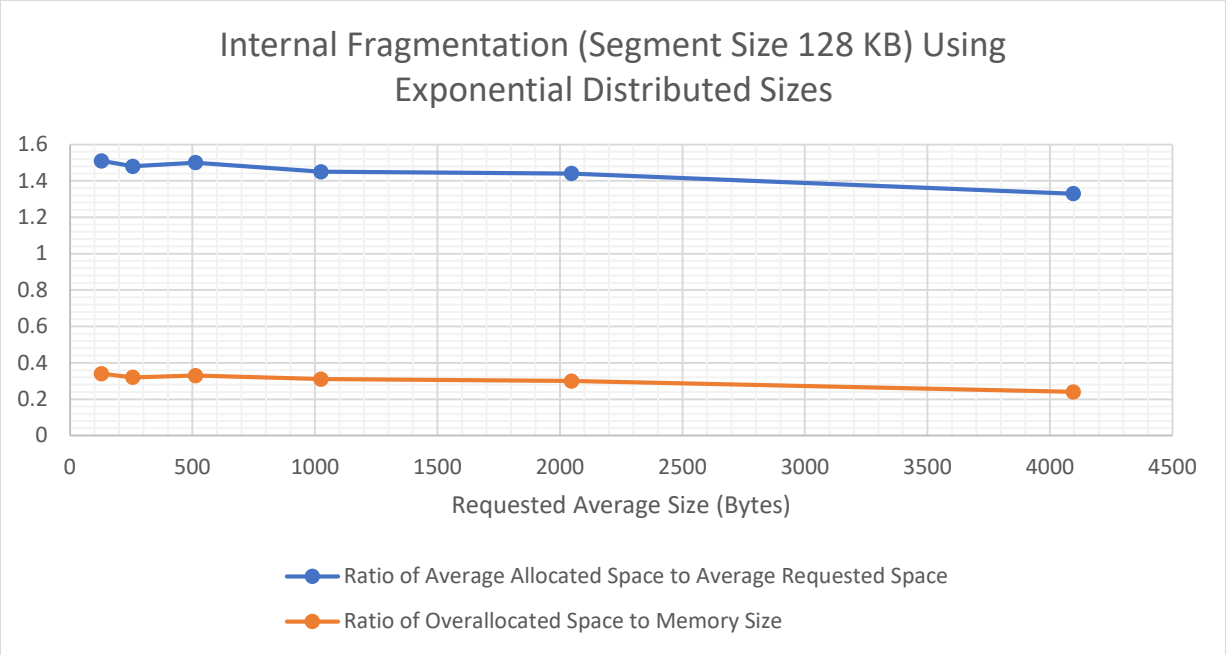

Figure 3. Internal Fragmentation (Segment Size 128 KB) Using Exponential Distributed Sizes

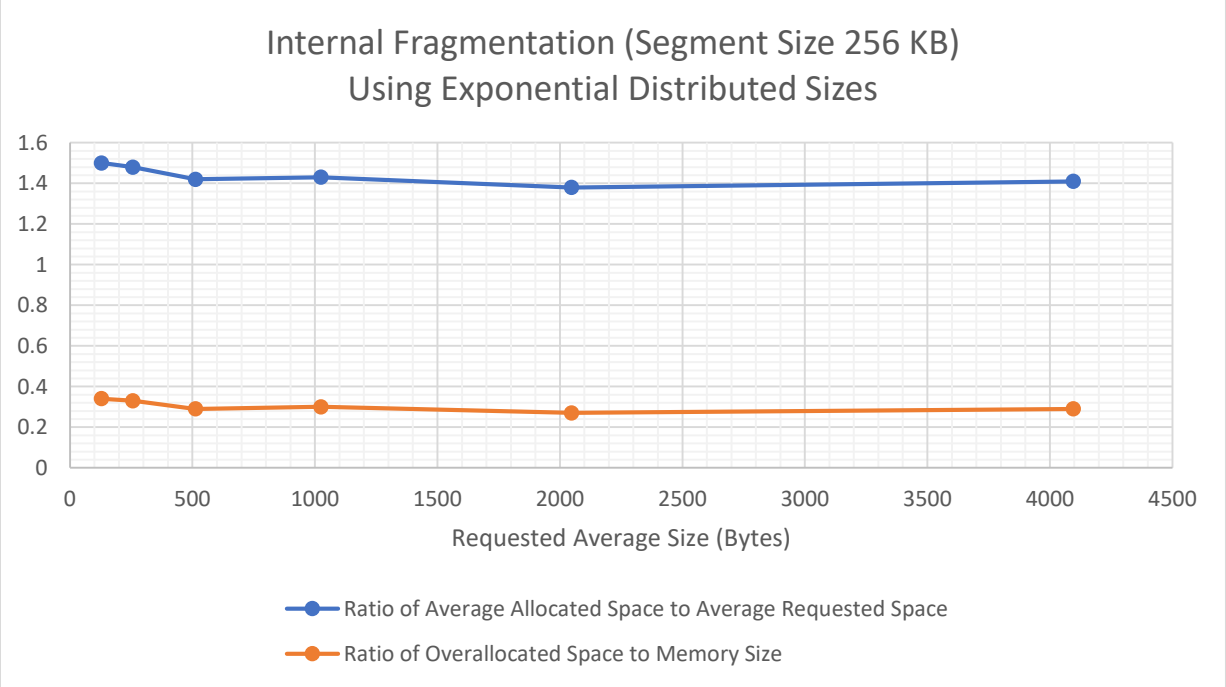

Figure 4. Internal Fragmentation (Segment Size 256 KB) Using Exponential Distributed Sizes

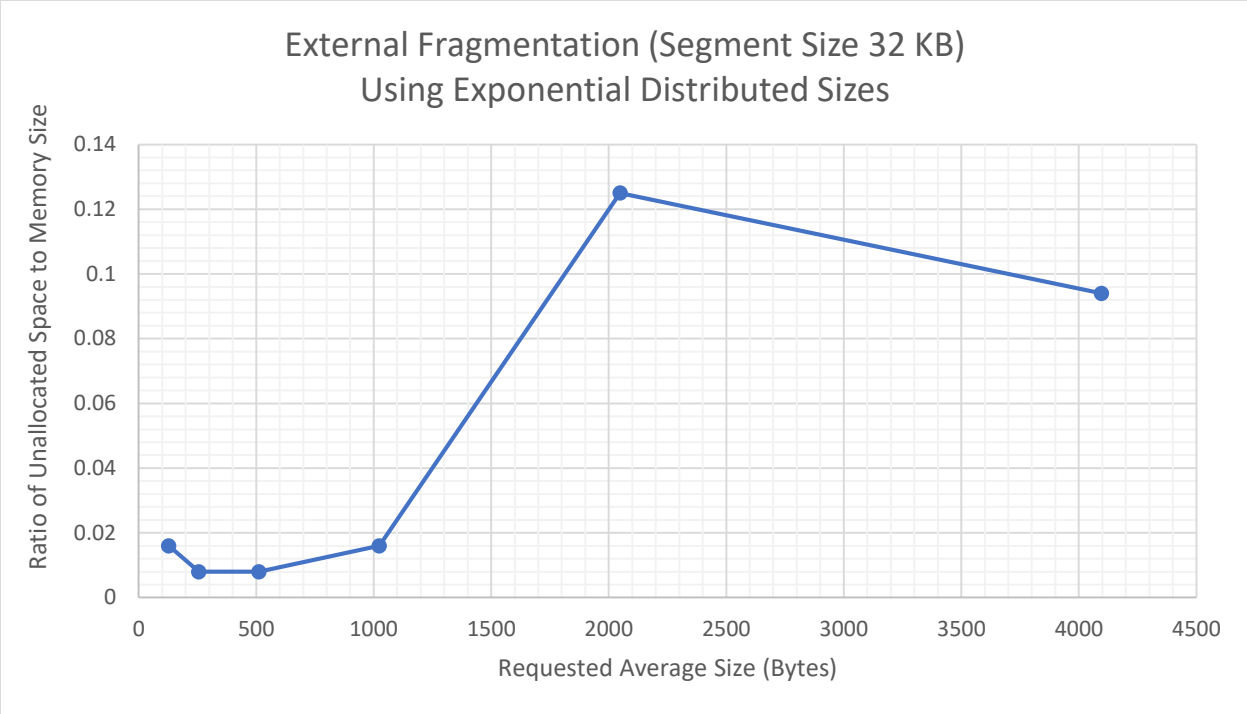

Figure 5. External Fragmentation (Segmentation Size 32 KB) Using Exponential Distributed Sizes

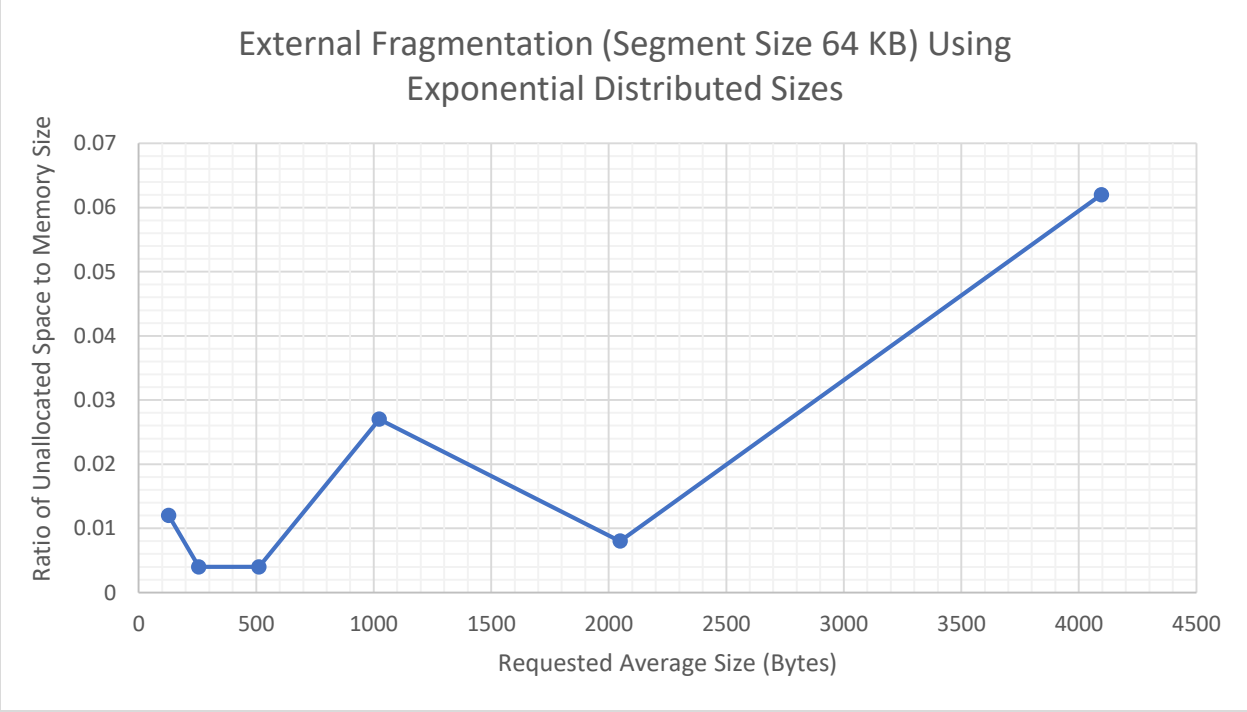

Figure 6. External Fragmentation (Segment Size 64 KB) Using Exponential Distributed Sizes

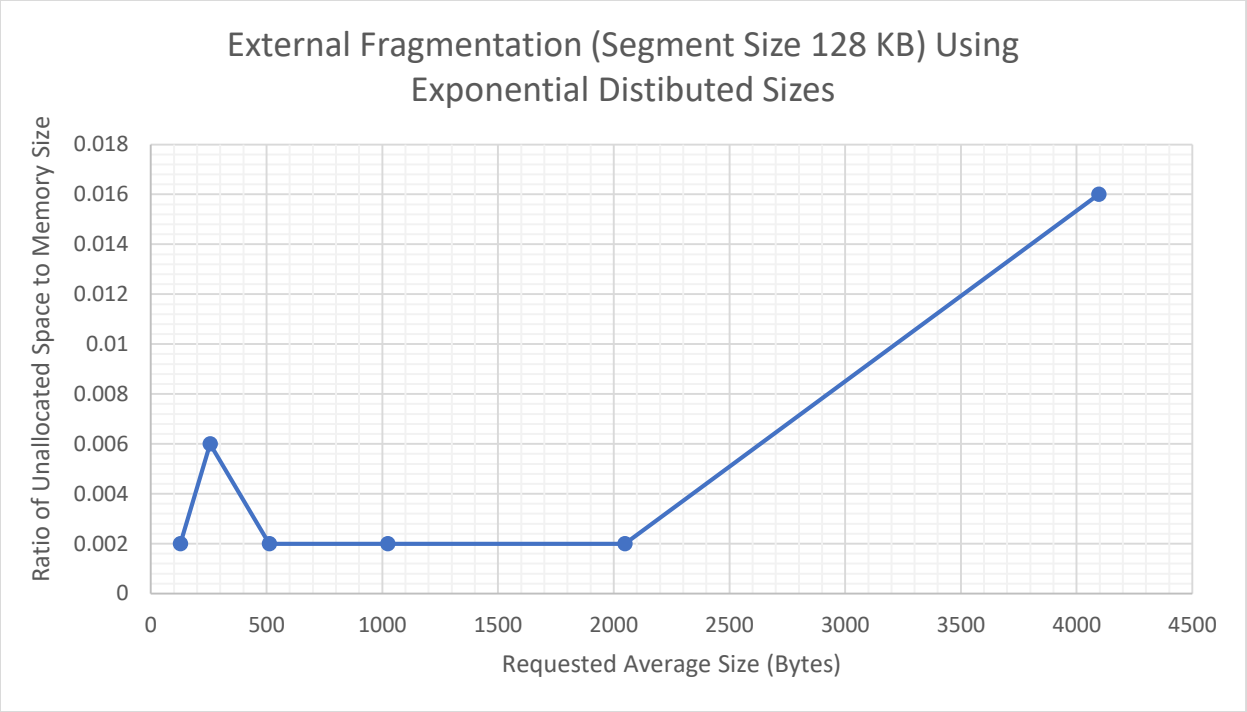

Figure 7. External Fragmentation (Segment Size 128 KB) Using Exponential Distributed Sizes

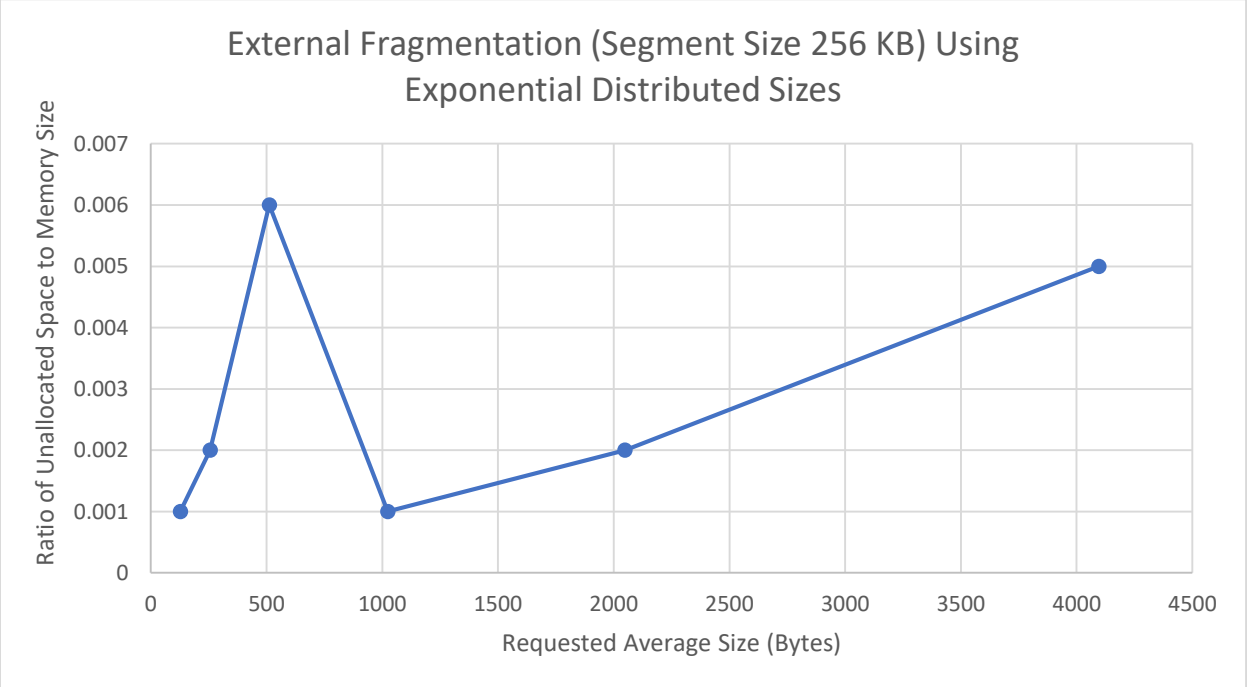

Figure 8. External Fragmentation (Segment Size 256 KB) Using Exponential Distributed Sizes

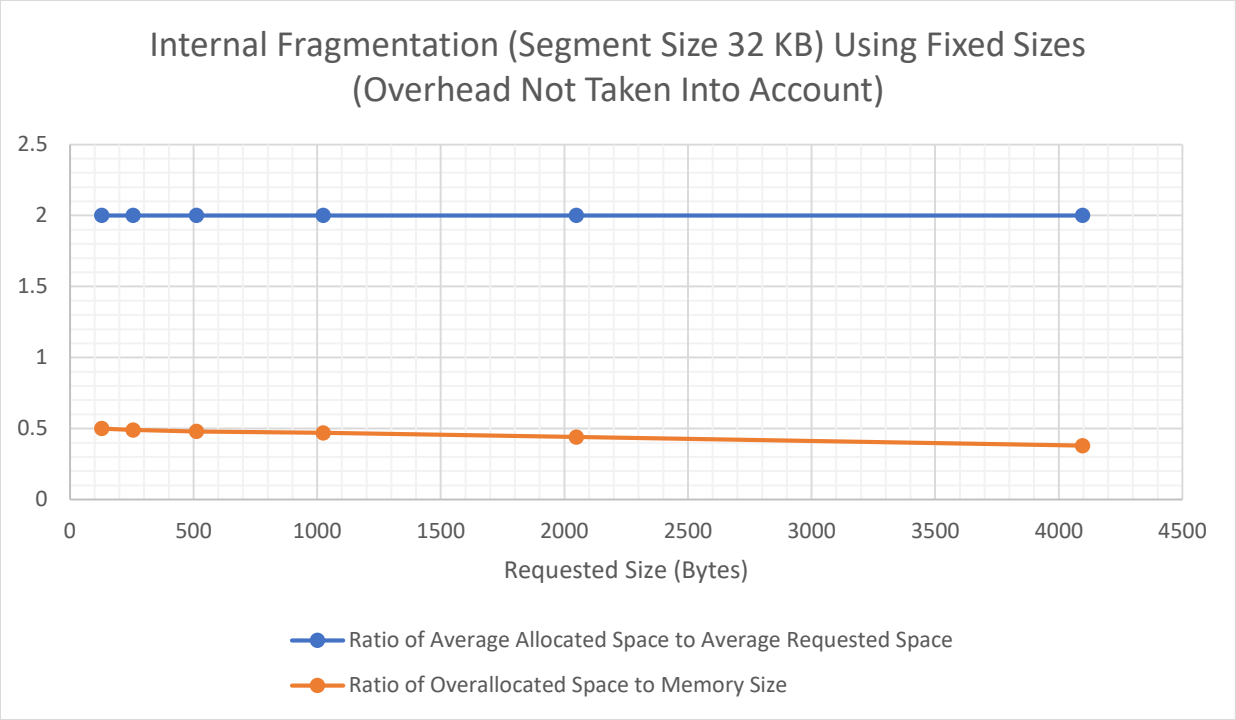

Figure 9. Internal Fragmentation (Segment Size 32 KB) using fixed sizes (overhead not taken into account)

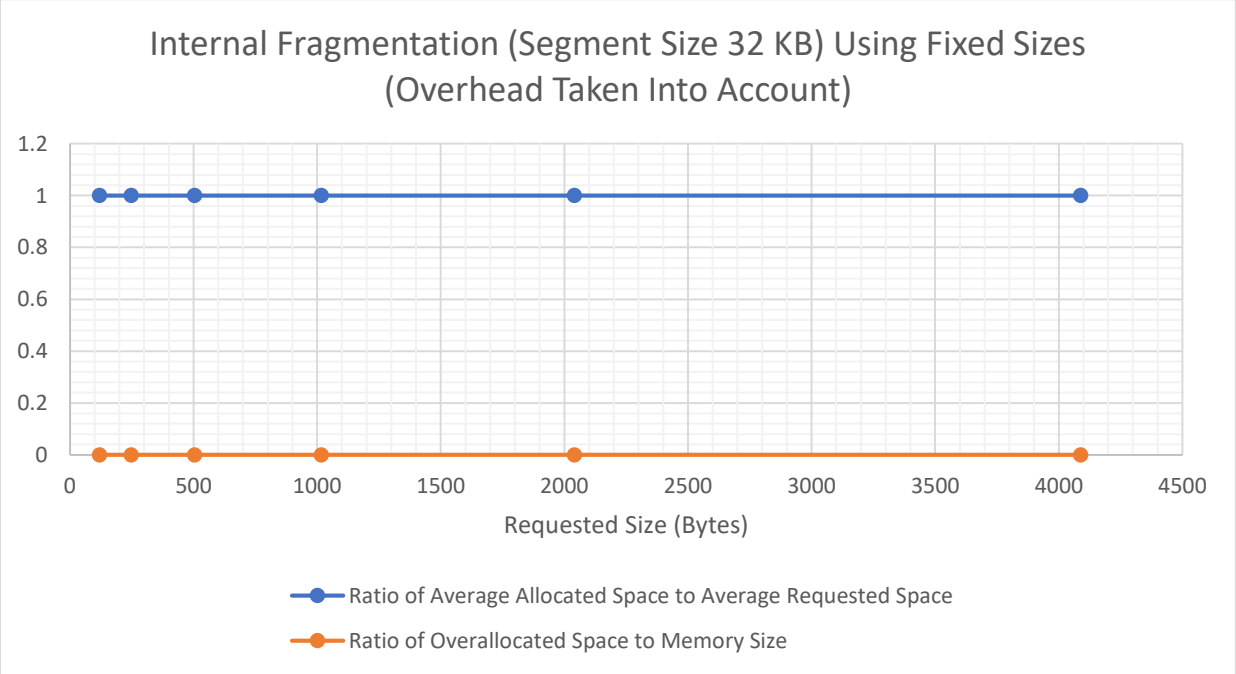

Figure 10. Internal Fragmentation (Segment Size 32 KB) using fixed sizes (overhead taken into account)

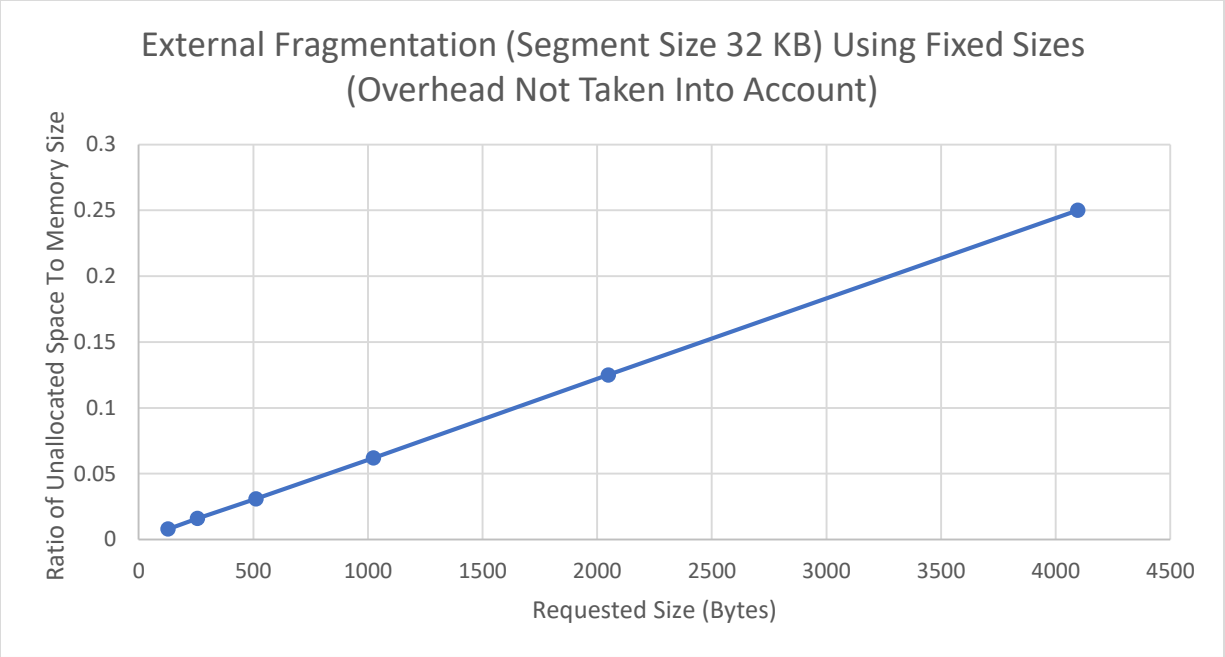

Figure 11. External Fragmentation (Segment Size 32 KB) using fixed sizes (overhead not taken into account)

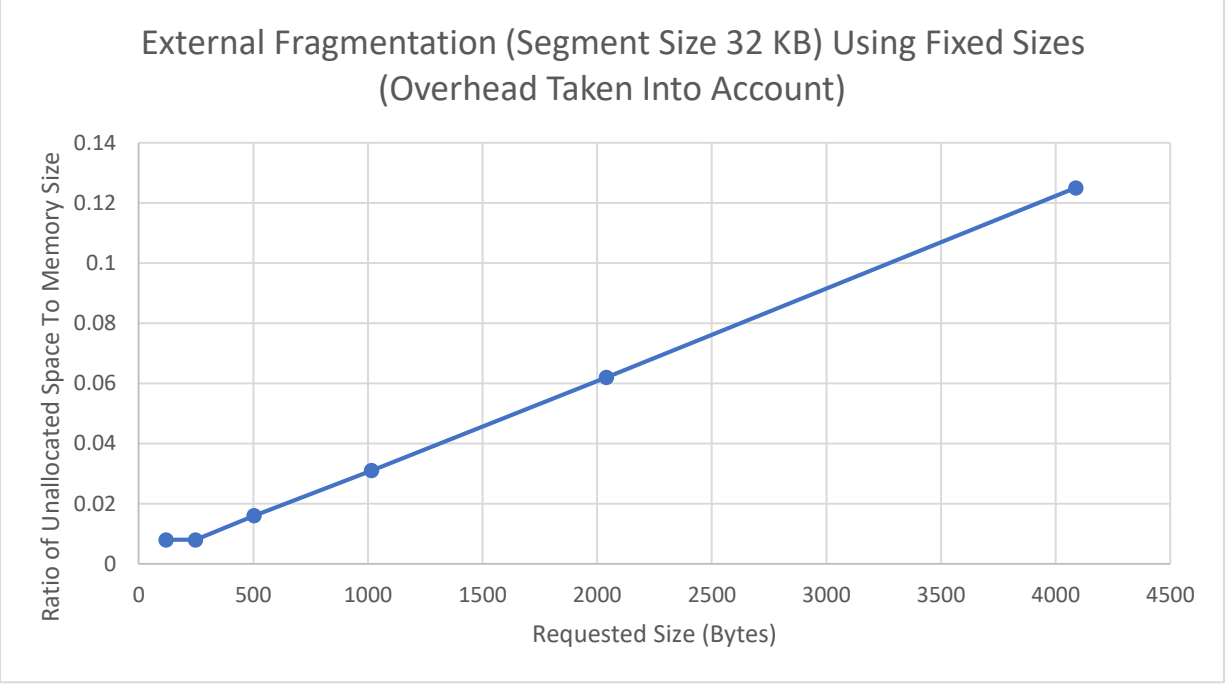

Figure 12. External Fragmentation (Segment Size 32 KB) using fixed sizes (overhead taken into account)

3. Analysis

Internal fragmentation was measured using two different approaches. The first approach was measuring the ratio of average allocated space to average requested space. According to our results, when the requested sizes were exponentially distributed with an average, the average increased the internal fragmentation decreased. The reason for this was, as the average size increased, fewer blocks could be allocated. The average allocated space was not high compared to the cases where the average requested size was small. According to our results, when the requested sizes were fixed and overhead was not taken into account, the measure for internal fragmentation was 2 for different requested sizes. This means that the buddy algorithm allocates exactly twice the memory requested. The internal fragmentation was higher compared to the case where requested sizes were exponentially distributed. When the requested sizes were fixed, and overhead was taken into account, the measure for internal fragmentation was 1 for different sizes. This means that the buddy algorithm allocates only 8 bytes more than the requested size, which is very efficient. These trends were similar for different segment sizes.

In internal fragmentation, the second approach was the ratio of over-allocated space to memory size. According to our results, when the requested sizes were exponentially distributed with an average, as the average increased, the internal fragmentation decreased. The reason for this was, as the average size increased, fewer blocks could be allocated. When the requested sizes were fixed, and overhead was not taken into account, the measure of internal fragmentation was around 0.5 for different requested sizes. This means that we could only use half of the memory and waste the other half. When the requested sizes were fixed and overhead was taken into account, internal fragmentation was always 0 for different requested sizes. In this case, the memory was used efficiently. These trends were similar for different segment sizes.

In external fragmentation, we measured the ratio of free memory to total memory when a requested allocation cannot be made because no block can handle that request, although the total free memory is enough to handle the request. When the requested sizes have exponential distribution, external fragmentation increases when the requested average size increases. This trend was similar for different segment sizes. This can be because the requested average block size increased the memory was divided into fewer fragments, so the memory usage is low. So, the free space increased when the external fragmentation increased.

When the requested sizes were fixed and overhead was not taken into account, external fragmentation was measured linearly for different requested sizes. As the requested size increased, the memory was divided into fewer fragments, so the memory usage is low. So, the free space increased when the external fragmentation increased. The same trend occurred when the requested sizes were fixed and overhead was taken into account—these patterns for similar for different segment sizes.

Reference

J. L. Peterson and T. A. Norman, “Buddy Systems,” Communications of the ACM, Vol. 20, No. 6, pp. 421–431, Jun. 1977.

Appendix:

A Random Sizes

Experiment program to measure internal and external fragmentation for the segment sizes of 32 KB, 64 KB, 128 KB, and 256 KB using exponentially distributed request sizes.

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "sbmem.h"
#include <time.h>


int powerOf2(int n)
{
    return (int)log2(n - 1) + 1;
}

int main()
{
    srand(time(NULL));
    for (int k = 0; k < 4; k++) {
        int segmentSize = 32768 * (1 << k);
        printf("\nSEGMENT SIZE=%d\n", segmentSize);

        for (int j = 0; j < 6; j++) {
            int allocSize = 128 * (1 << j);
            int exp;
            sbmem_remove();
            sbmem_init(segmentSize);
            int ret = sbmem_open();
            init_experiment();
            open_experiment();
            if (ret != -1) {
                while (1) {
                    exp = (int) (log(1-((float)rand()/RAND_MAX))/
                                (-1.0/allocSize));
                    while((exp > 4096) || (exp < 128)){
                        exp = (int) (log(1-((float)rand()/RAND_MAX))/
                                (-1.0/allocSize));
                    }

                    void* p = sbmem_alloc(exp);
                    if (p == NULL && (exp <= get_free_space())) {
                        break;
                    }
                }
```

```
                print_state();
                ret = sbmem_close();
            }
            else {
                printf("Could not open shared memory\n");
            }
            close_experiment();
        }
    }
    sbmem_remove();
    return (0);
}
```

B Fixed Sizes

Experiment program to measure internal and external fragmentation for the segment sizes of 32 KB, 64 KB, 128 KB, and 256 KB using fixed sizes to be requested.

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "sbmem.h"
#include <time.h>

int powerOf2(int n)
{
    return (int)log2(n - 1) + 1;
}

int main()
{
    for (int k = 0; k < 4; k++) {
        int segmentSize = 32768 * (1 << k);
        printf("\nSEGMENT SIZE=%d\n", segmentSize);
        for (int j = 0; j < 6; j++) {
            int allocSize = 128 * (1 << j);
            sbmem_remove();
            sbmem_init(segmentSize);
            int ret = sbmem_open();
            init_experiment();
            open_experiment();
            if (ret != -1) {
                int realAlloc = allocSize;
                if (powerOf2(allocSize + 8) != powerOf2(allocSize)) {
                    realAlloc = allocSize * 2;
                }
                for (int i = 0; i < (segmentSize / realAlloc) - 1; i++) {
                    void* p = sbmem_alloc(allocSize);
                    if (p == NULL) {
                        break;
                    }
                }
                print_state();
                ret = sbmem_close();
            }
            else {
                printf("Could not open shared memory\n");
            }
            close_experiment();
        }
    }
    sbmem_remove();
    return (0);
}
```