CS107, Lecture 22
Privacy and Trust + Heap Allocators

Reading: B&O 9.9, 9.11

Learning Goals

- Learn about the connections between privacy, security and trust
- Learn the restrictions, goals and assumptions of a heap allocator
- Understand the conflicting goals of utilization and throughput

Lecture Plan

- Privacy and Trust
- The heap so far
- What is a heap allocator?
- Heap allocator requirements and goals

Lecture Plan

- **Privacy and Trust**
- The heap so far
- What is a heap allocator?
- Heap allocator requirements and goals

Privacy and Trust

- Our learning about assembly and program execution helps us better understand computer security (the protection of data, devices, and networks from disruption, harm, theft, unauthorized access or modification).
- Computer security is important in part because it enables privacy.
- In understanding computer security, it's essential to understand the context in which it comes up (privacy and trust).

Data Breaches

Privacy/trust example: data breaches

- California list of data security breaches: link
- How does a data breach make a customer feel?

Privacy

What is privacy? 4 possible framings in two categories:

Individualist: the value of privacy as an individual right

- Privacy as **control of information** – controlling how our private information is shared with others.
- Privacy as **autonomy** – capacity to choose/decide for ourselves what is valuable.

Social: the value of privacy for a group

- Privacy as **social good** – social life would be unlivable without privacy.
- Privacy (protection) as based in **trust** – privacy enables trusting relationships

Privacy

Privacy as **control of information** – controlling how our information is communicated to others.

- Consent requires *free* choice with available alternatives and *informed* understanding of what is being offered.
- How many of you just skip past the terms of service for new online services you sign up for?
- Do you feel in control of your information with the services you choose to use? Why or why not? If you're working on a service, how can you respect privacy while achieving product goals?
- Control over personal data being collected (e.g. data exports from services you use, privacy dashboards, device privacy protections)

Privacy

Art. 1 GDPR

Subject-matter and objectives

1. This Regulation lays down rules relating to the protection of natural persons with regard to the processing of personal data and rules relating to the free movement of personal data.

2. This Regulation protects fundamental rights and freedoms of natural persons and in particular their right to the protection of personal data.

3. The free movement of personal data within the Union shall be neither restricted nor prohibited for reasons connected with the protection of natural persons with regard to the processing of personal data.

Media & Entertainment

Instagram launches "Data Download" tool to let you leave

Josh Constine / 9:44 AM PDT • April 24, 2018 Comment

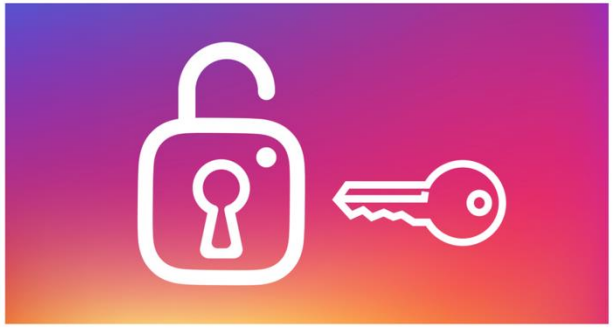


Image Credits: Bryce Durbin/TechCrunch /

TECH / APPLE / GOOGLE

Apple now lets you automatically transfer your iCloud Photo Library to Google Photos

/ Not everything can come along for the ride, though

By Mitchell Clark

Mar 3, 2021 at 1:10 PM PST

Google Report content on Google

Personal Data Removal Request Form

For privacy and data protection reasons (such as pursuant to the EU General Data Protection Regulation) you may have the right to ask for certain personal data relating to you to be removed.

This form is for requesting the removal of specific results for queries that include your name from Google Search. Google LLC is the controller responsible for the processing of personal data carried out in the context of determining the results shown by Google Search, as well as handling delisting requests sent through this form.

Privacy

Privacy as **autonomy** – capacity to choose/decide for ourselves what is valuable.

- Links to autonomy over our own lives and our ability to lead them as we choose.
- Do you feel that your autonomy is always respected when using products and services? Why or why not?

"[P]rivacy is valuable because it acknowledges our respect for persons as autonomous beings with the capacity to love, care and like—in other words, persons with the potential to freely develop close relationships" (Innes 1992)

Individualist Models of Privacy

Privacy as **autonomy** and privacy as **control over information** focus the value of privacy at an individual level.

- Individual privacy can conflict with interests of society or the state.
- Many debates over ”privacy vs. security” – whether one should be sacrificed for the other
  - Apple v. FBI case re: unlocking iPhones (link)
  - Debates around encryption (link)
- Where do your beliefs fall in balancing privacy and security? When (if at all) is it ok to sacrifice one, and how much?

Privacy

Privacy as **social good** – social life would be unlivable without privacy.

- Privacy has a social value in bringing about the kind of society we want to live in.
- What would society look like without privacy?

Privacy

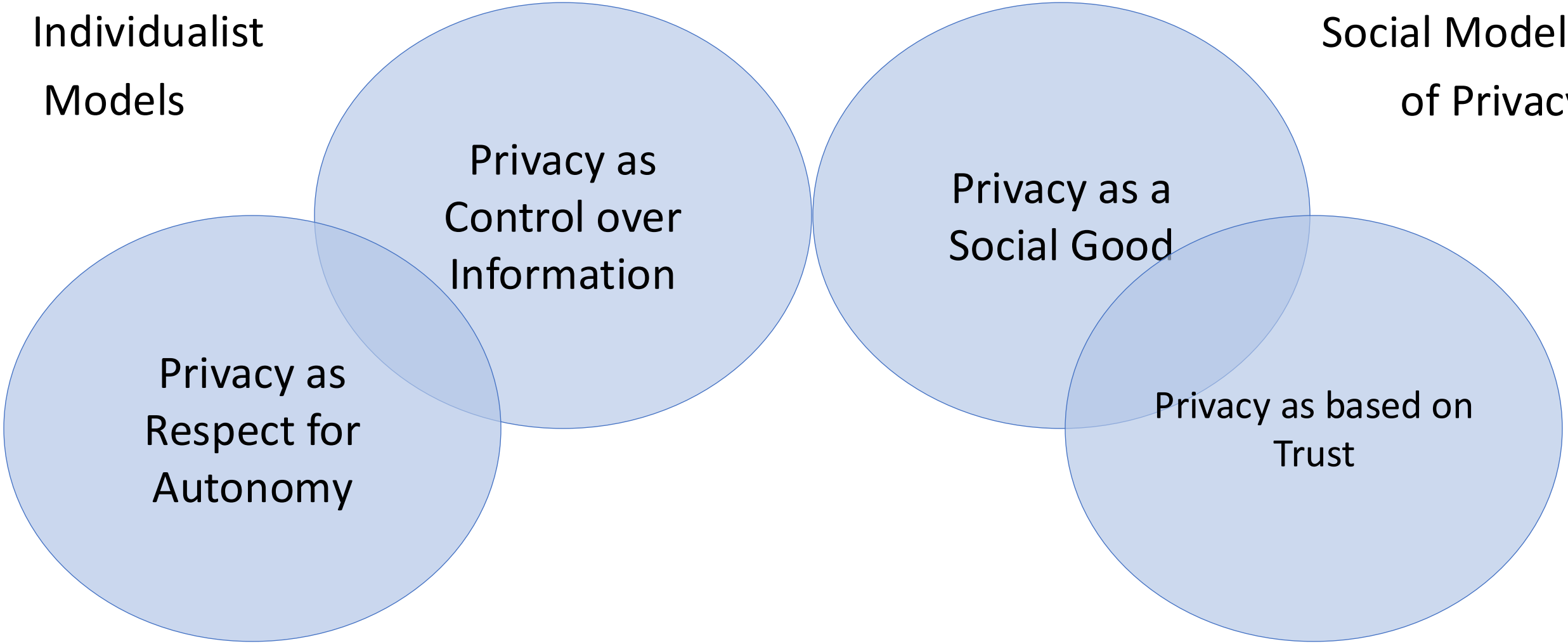
Privacy (protection) as based in **trust** – privacy enables trusting relationships

- Privacy may help enable trusting relationships essential for cooperation.
  - For instance, a *fiduciary*: someone who stands in a legal or ethical relationship of trust with another person (or group). The fiduciary must act for the benefit of and in the best interest of the other person. E.g. tax filer with access to your bank account
    - Should anyone who has access to personal info have a *fiduciary* responsibility? (Richards & Hartzog 2020).
- This model of privacy stresses the essential relationship of trust placed in any holder of personal data and the responsibilities that result from this trust.

Models of Privacy

Individualist Models

Social Models of Privacy



Who Should We Trust?

Both security and privacy rely on trusted people (who administer security, perform penetration tests, submit vulnerabilities to databases, or keep private information secret). The final piece of the security puzzle is understanding trust.

Trust = Reliance + Risk of Betrayal

What makes trust unique to relationships between people is that trust exposes one to being *betrayed or being let down* (Baier 1986).

Penetration Testing & Trust

Penetration testing is the practice of encouraging or hiring security researchers / contractors to find vulnerabilities in one's own code or system.

- Position of trust – tester is given access to the system and encouraged to find exploitable vulnerabilities, expected to share what they have found with you.
- Means *relying on* their skill at finding vulnerabilities and *trusting* that their ethical compass will lead them to tell you and to act as a trustworthy *fiduciary* (guardian of your interests).

In Assignment 5, you have the opportunity to explore this further!

Loss of Privacy

Loss of privacy can cause us various harms, including:

- ***Aggregation*:** combining personal information from various sources to build a profile of someone
- ***Exclusion:*** not knowing how our information is being used, or being unable to access or modify it (Google removing personal info from search – link)
- ***Secondary Use***: using your information for purposes other than what was intended without permission.

Mitigation: Differential Privacy

Differential privacy is a formal measure of privacy for datasets to try and protect individuals from aggregation by making them harder to identify (Dwork 2008).

- Imagine a large database, e.g., a medical database, with personal information and records of past activity tied to a name.
- The records might be useful for research purposes, or to train a machine learning model to predict future health outcomes, but what if giving access to the records exposed the privacy of individual person's health records?
- Differential privacy adds inconsequential noise (e.g., changing a birthday from 2001 to 2002) or removes records to make individuals harder to identify while preserving the utility of the dataset overall.

Trust Models

In every evaluation of privacy, we can ask: who is trusted? Who is distrusted? Does this model concentrate trust (and therefore power) in a single individual or small group, or does it distribute trust?

Differential Privacy's Trust Model

Differential privacy assumes that the only threat to privacy is an *external user querying the database* who must be prevented from aggregating data that could identify a user.

- In other words, the *trust model* of differential privacy is that the database owners and maintainers are to be fully trusted, and no one else.
- But is that the only threat? Differential privacy does not protect against improper use by people with full access to data or against leaks of the whole database, which may be the primary data exposure risks.

Differential privacy also does not question the assumption that amassing & storing large amounts of personal data is worth the risk of inevitable leaks (Rogaway 2015).

Lecture Plan

- Privacy and Trust
- **The heap so far**
- What is a heap allocator?
- Heap allocator requirements and goals

CS107 Topic 6: How do the core `malloc`/`realloc`/`free` memory-allocation operations work?

CS107 Topic 6

How do the core malloc/realloc/free memory-allocation operations work?

Why is answering this question important?

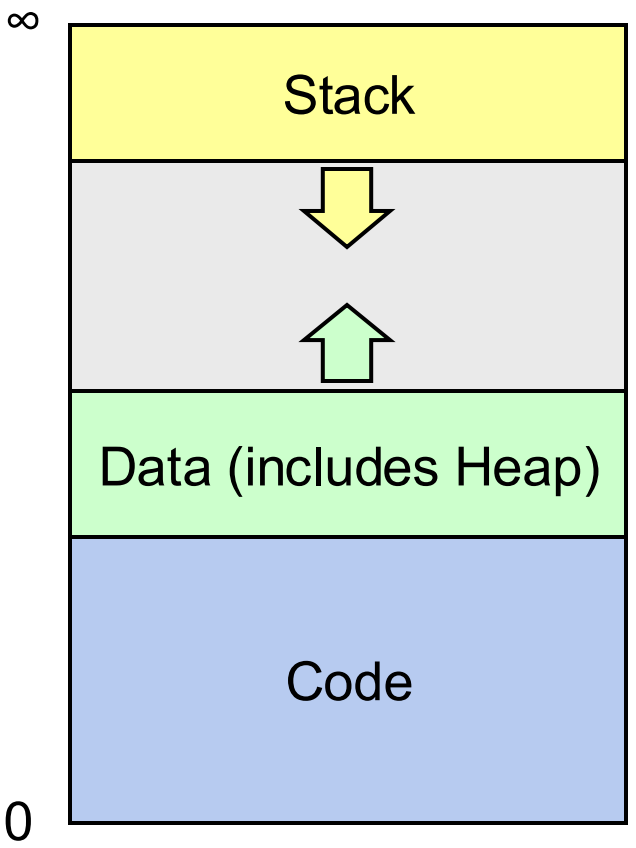
- Combines techniques from across the quarter (bits/bytes, pointers, memory, generics, assembly, efficiency, testing, and more) to understand a real-world system that you have relied on all quarter!
- Learning about the design and tradeoffs in a real-world large system gives us a great example of how to evaluate different designs when there's no one "right" answer.

assign6: implement two different possible designs for a heap allocator, implementing malloc/realloc/free.

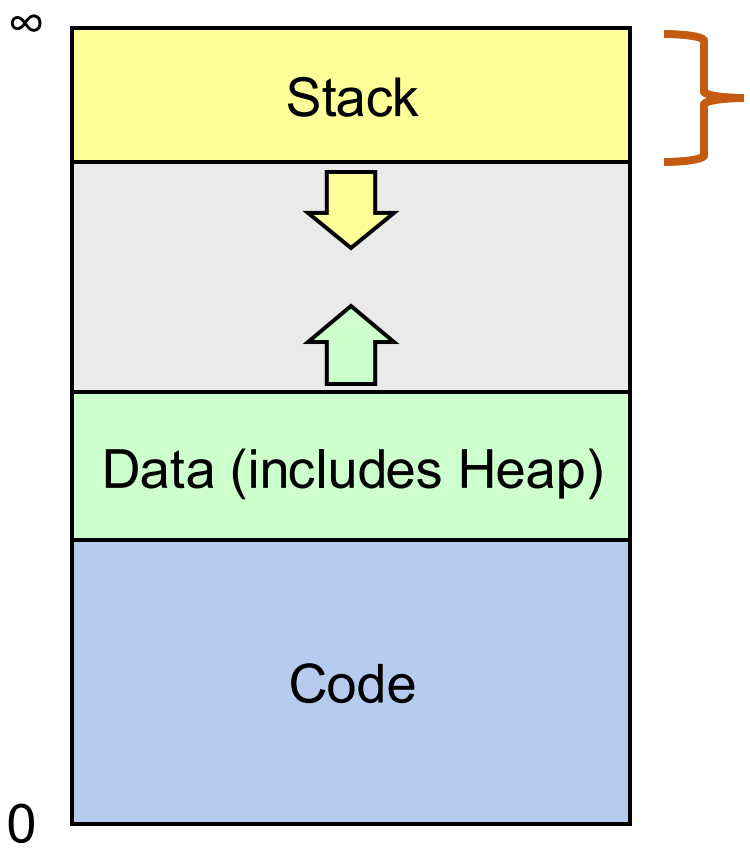
Running a program

- **Creates new process**
- **Sets up address space/segments**
- **Read executable file, load instructions, global data**
  Mapped from file into gray segments
- **Libraries loaded on demand**

- **Set up stack**
  Reserve stack segment, init %rsp, call main
- **malloc written in C, will init self on use**
  Asks OS for large memory region,
  parcels out to service requests



The Stack

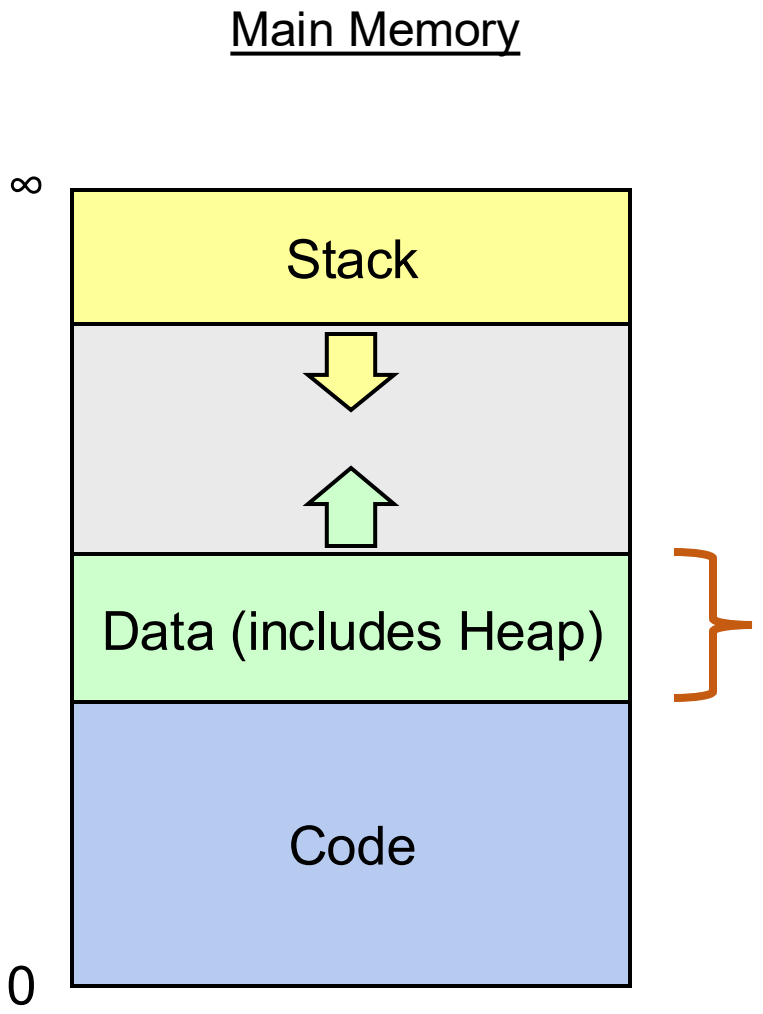


Stack memory "goes away" after function call ends.

Automatically managed at compile-time by gcc

From Assembly:
Stack management == moving %rsp around (pushq, popq, mov)

Today: The Heap



Heap memory persists until caller indicates it no longer needs it.

Managed by C standard library functions (malloc, realloc, free)

This lecture:
How does heap management work?

Lecture Plan

- Privacy and Trust
- The heap so far
- **What is a heap allocator?**
- Heap allocator requirements and goals

Your role so far: Client

```
void *malloc(size_t size);
```

Returns a pointer to a block of heap memory of at least size bytes, or NULL if an error occurred.

```
void free(void *ptr);
```

Frees the heap-allocated block starting at the specified address.

```
void *realloc(void *ptr, size_t size);
```

Changes the size of the heap-allocated block starting at the specified address to be the new specified size. Returns the address of the new, larger allocated memory region.

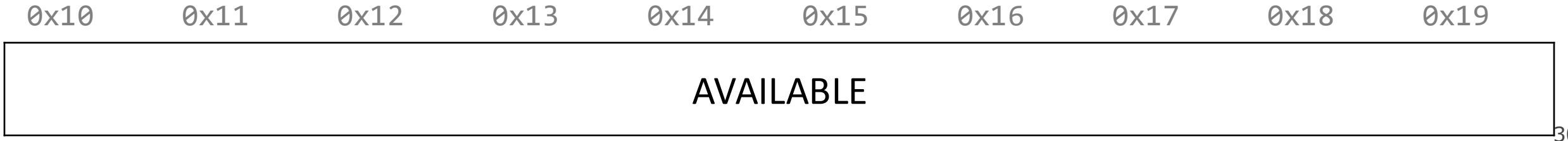
Your role now: Heap Hotel Concierge

(aka **Heap Allocator**)

http://screencrave.com/wp-content/uploads/2014/03/the-grand-budapest-hotel-anderson-image-2.jpg

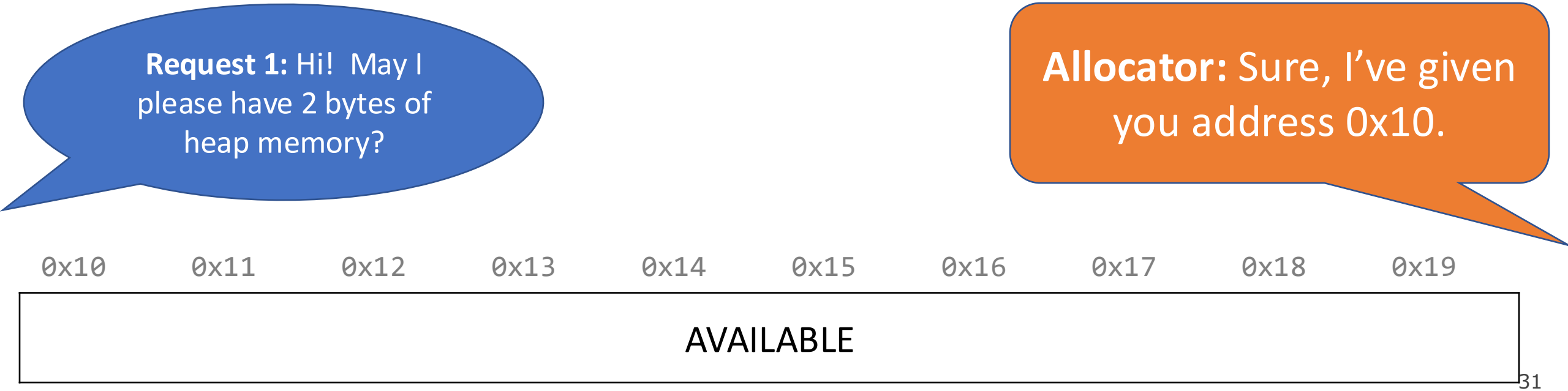
What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).



What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

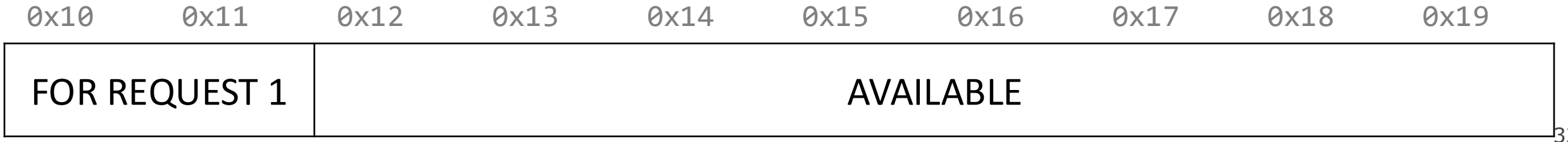


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 1: Hi! May I please have 2 bytes of heap memory?

Allocator: Sure, I've given you address 0x10.

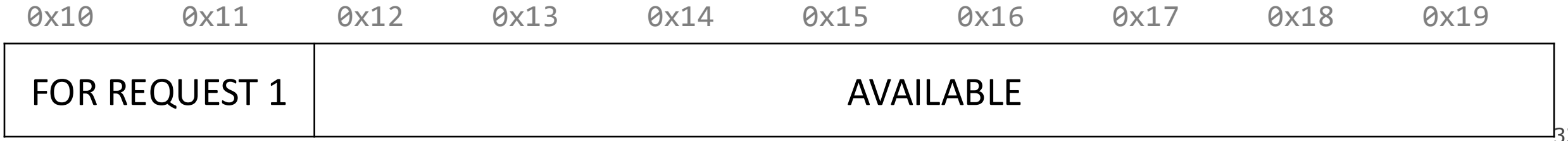


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 2: Howdy! May I please have 3 bytes of heap memory?

Allocator: Sure, I've given you address 0x12.

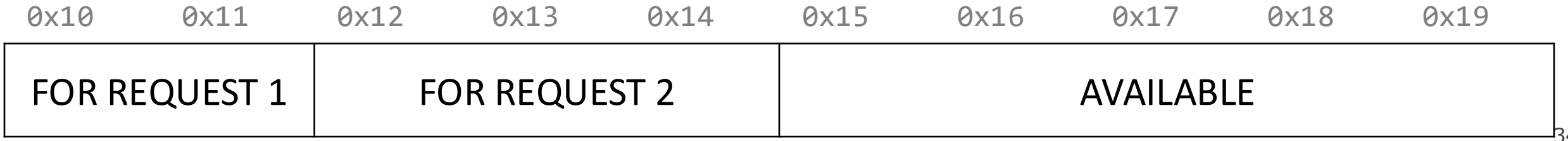


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 2: Howdy! May I please have 3 bytes of heap memory?

Allocator: Sure, I've given you address 0x12.

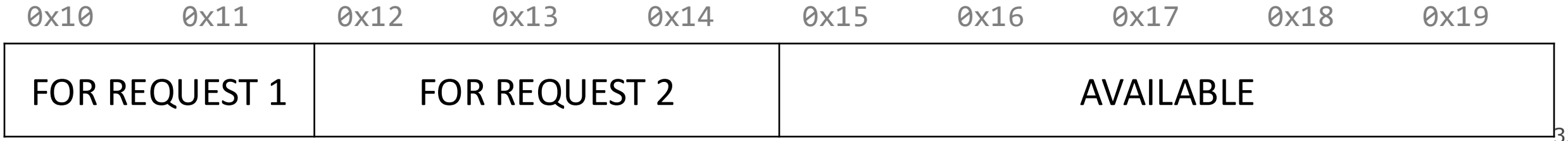


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 1: I'm done with the memory I requested. Thank you!

Allocator: Thanks. Have a good day!

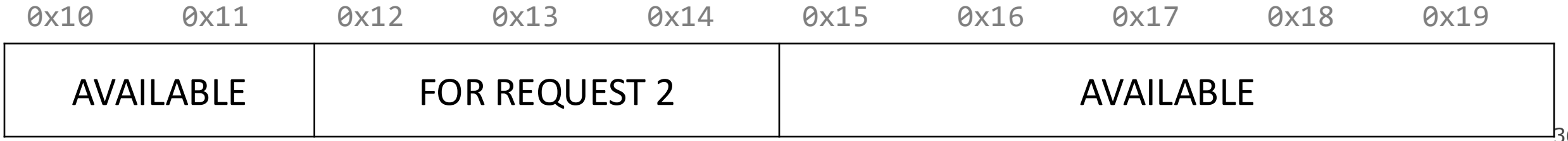


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 1: I'm done with the memory I requested. Thank you!

Allocator: Thanks. Have a good day!

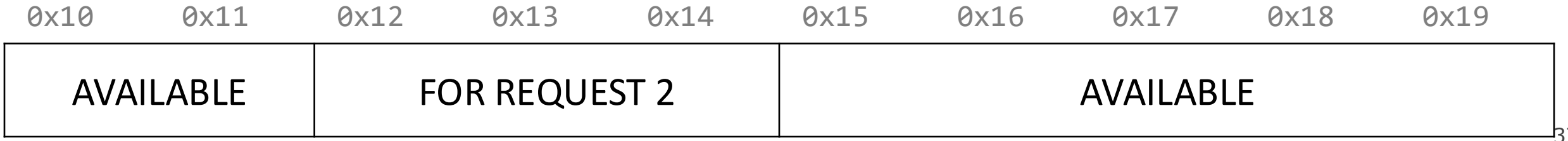


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 3: Hello there! I'd like to request 2 bytes of heap memory, please.

Allocator: Sure thing. I've given you address 0x10.

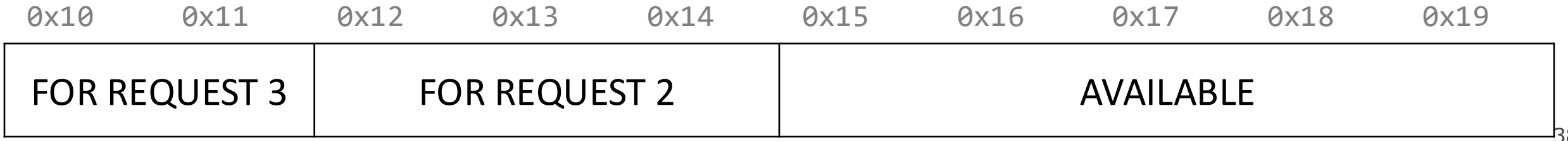


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 3: Hello there! I'd like to request 2 bytes of heap memory, please.

Allocator: Sure thing. I've given you address 0x10.

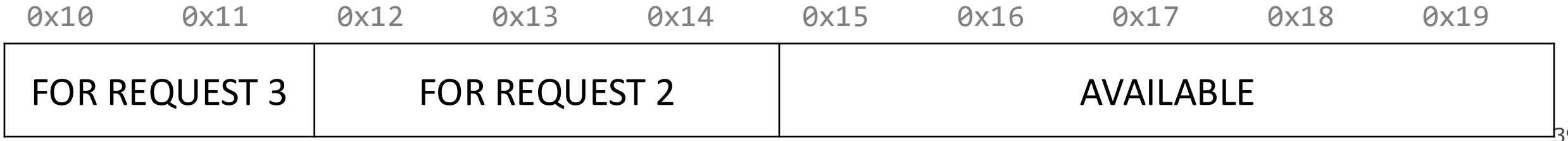


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 3: Hi again! I'd like to request the region of memory at 0x10 be reallocated to 4 bytes.

Allocator: Sure thing. I've given you address 0x15.

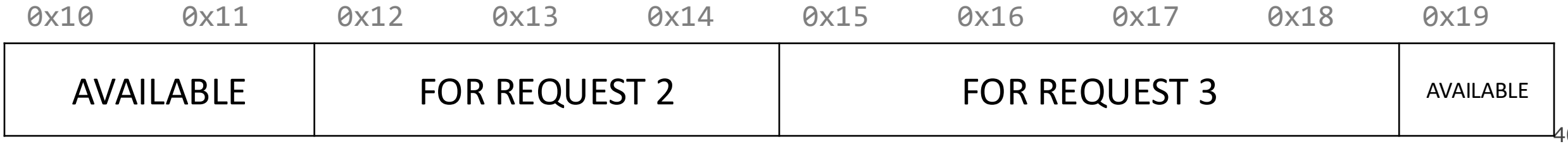


What is a heap allocator?

- A heap allocator is a set of functions that fulfills requests for heap memory.
- On initialization, a heap allocator is provided the starting address and size of a large contiguous block of memory (the heap).
- A heap allocator must manage this memory as clients request or no longer need pieces of it.

Request 3: Hi again! I'd like to request the region of memory at 0x10 be reallocated to 4 bytes.

Allocator: Sure thing. I've given you address 0x15.



Lecture Plan

- Privacy and Trust
- The heap so far
- What is a heap allocator?
- **Heap allocator requirements and goals**

Heap Allocator Functions

```
void *malloc(size_t size);

void free(void *ptr);

void *realloc(void *ptr, size_t size);
```

Heap Allocator Requirements

A heap allocator must…

1. Handle arbitrary request sequences of allocations and frees
2. Keep track of which memory is allocated and which is available
3. Decide which memory to provide to fulfill an allocation request
4. Immediately respond to requests without delay
5. Return addresses that are 8-byte-aligned (must be multiples of 8).

Heap Allocator Requirements

A heap allocator must…

1. **Handle arbitrary request sequences of allocations and frees**
2. Keep track of which memory is allocated and which is available
3. Decide which memory to provide to fulfill an allocation request
4. Immediately respond to requests without delay
5. Return addresses that are 8-byte-aligned (must be multiples of 8).

A heap allocator cannot assume anything about the order of allocation and free requests, or even that every allocation request is accompanied by a matching free request.

Heap Allocator Requirements

A heap allocator must…

1. Handle arbitrary request sequences of allocations and frees
2. **Keep track of which memory is allocated and which is available**
3. Decide which memory to provide to fulfill an allocation request
4. Immediately respond to requests without delay
5. Return addresses that are 8-byte-aligned (must be multiples of 8).

A heap allocator marks memory regions as **allocated** or **available**. It must remember which is which to properly provide memory to clients.

Heap Allocator Requirements

A heap allocator must…

1. Handle arbitrary request sequences of allocations and frees
2. Keep track of which memory is allocated and which is available
3. **Decide which memory to provide to fulfill an allocation request**
4. Immediately respond to requests without delay
5. Return addresses that are 8-byte-aligned (must be multiples of 8).

A heap allocator may have options for which memory to use to fulfill an allocation request. It must decide this based on a variety of factors.

Heap Allocator Requirements

A heap allocator must…

1. Handle arbitrary request sequences of allocations and frees
2. Keep track of which memory is allocated and which is available
3. Decide which memory to provide to fulfill an allocation request
4. **Immediately respond to requests without delay**
5. Return addresses that are 8-byte-aligned (must be multiples of 8).

A heap allocator must respond immediately to allocation requests and should not e.g. prioritize or reorder certain requests to improve performance.

Heap Allocator Requirements

A heap allocator must…

1. Handle arbitrary request sequences of allocations and frees
2. Keep track of which memory is allocated and which is available
3. Decide which memory to provide to fulfill an allocation request
4. Immediately respond to requests without delay
5. **Return addresses that are 8-byte-aligned (must be multiples of 8).**

Heap Allocator Goals

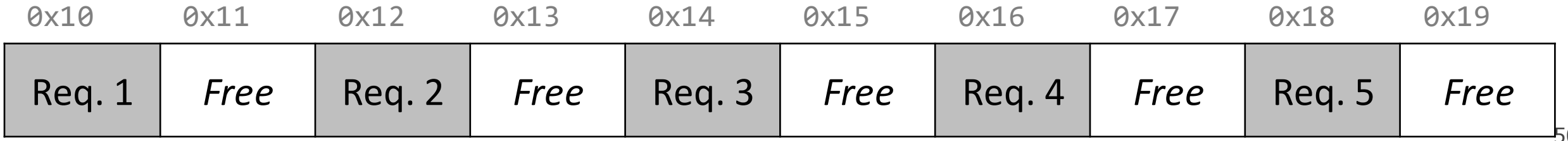
- Goal 1: Maximize **throughput**, or the number of requests completed per unit time. This means minimizing the average time to satisfy a request.
- Goal 2: Maximize memory **utilization**, or how efficiently we make use of the limited heap memory to satisfy requests.

Utilization

- The primary cause of poor utilization is **fragmentation**. **Fragmentation** occurs when otherwise unused memory is not available to satisfy allocation requests.
  - **External Fragmentation (this example)**: no single space is large enough to satisfy a request, even though enough aggregate free memory is available
  - **Internal Fragmentation**: space allocated for a block is larger than needed (more later).
- In general: we want the largest address used to be as low as possible.

Request 6: Hi! May I please have 4 bytes of heap memory?

Allocator: I'm sorry, I don't have a 4 byte block available…

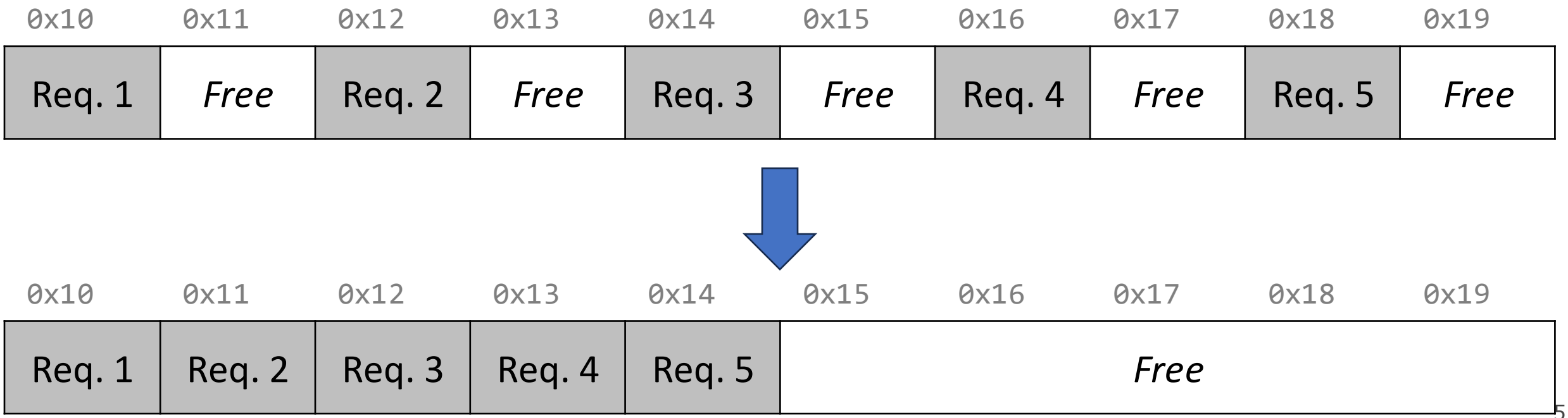


Utilization

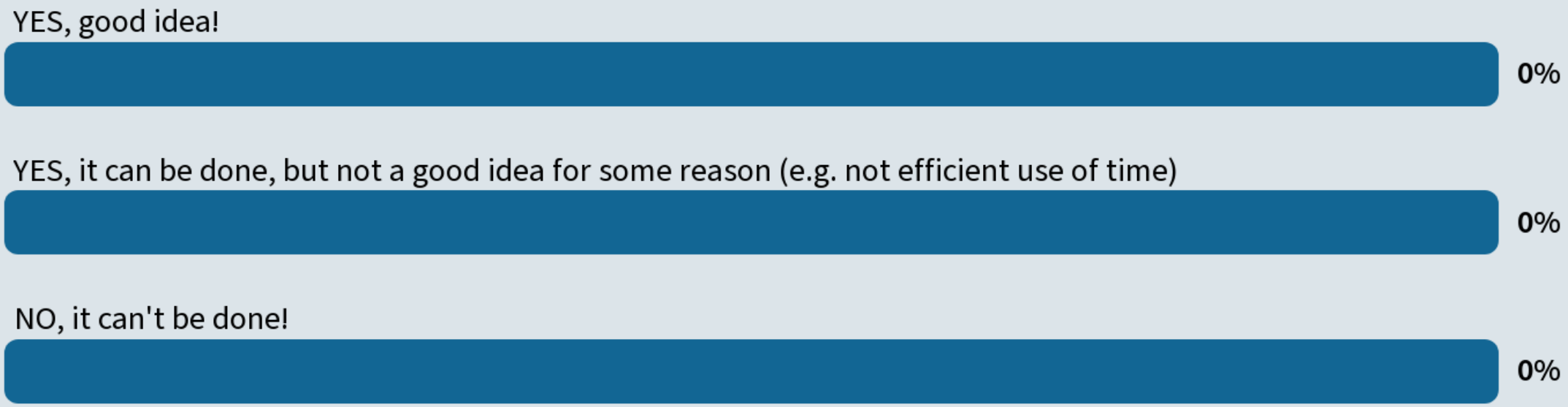
Question: Can we / should we shift these blocks down to make more space?

- YES, good idea!
- YES, but not a good idea!
- NO, it can't be done!

Respond on PollEv: pollev.com/cs107
or text CS107 to 22333 once to join.



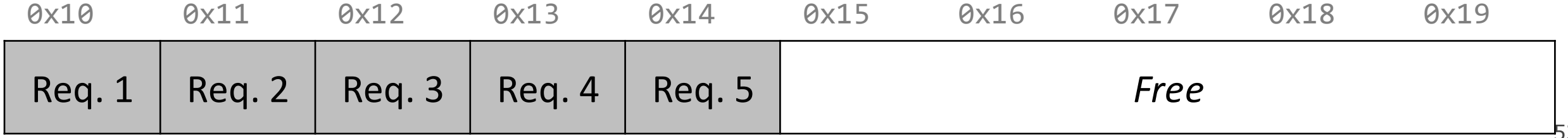
Can we shift these blocks down to make more space?



Utilization

Question: Can we / should we shift these blocks down to make more space?

- **No -** we have already guaranteed these addresses to the client. We cannot move allocated memory around, since this will mean the client will now have incorrect pointers to their memory!



Heap Allocator Goals

- Goal 1: Maximize **throughput**, or the number of requests completed per unit time. This means minimizing the average time to satisfy a request.
- Goal 2: Maximize memory **utilization**, or how efficiently we make use of the limited heap memory to satisfy requests.

These are seemingly conflicting goals – for instance, it may take longer to better plan out heap memory use for each request. **Heap allocators must find an appropriate balance between these two goals!**

Recap

- Privacy and Trust
- The heap so far
- What is a heap allocator?
- Heap allocator requirements and goals

Lecture 22 takeaways: Computer security comes up in discussions of privacy (individualist and social framings) and trust (reliance + risk of betrayal). How can we balance privacy and security? How can we mitigate potential harms?

A heap allocator is a set of functions that fulfills requests for heap memory. Seemingly-conflicting goals of maximizing throughput and memory utilization!