

# Coding Theory

CSE 445/545

January 30, 2023

# Let's do some introductions

Atri Rudra

319 Davis Hall

[atri@buffalo.edu](mailto:atri@buffalo.edu)

645-2464

Office hours: Mon 11:30am-12:20pm

# Your awesome TA: Ben Siegel



Office hours: Fill in the piazza poll!

NO office hours this week

Lectures will be videotaped



Still take notes!

# Handouts for today

## Syllabus

Linked from the course webpage

## Feedback polls

Up on piazza

# Plug for feedback polls

Completing the polls is voluntary & anonymous

## Purpose of the polls

For me to get an idea of your technical background

# One Stop Shop for the course

CSE 4/545 Syllabus Piazza Schedule Homeworks ▾ Mini Project ▾ Autolab Book

## CSE 445/545: Coding Theory



Spring 2023

<https://cse.buffalo.edu/faculty/atri/courses/coding-theory/webpage/spr23/>

### Under Construction

This page is still under construction. In particular, nothing here is final while this sign still remains here.

### CSE 4/545 events

Today			Jan 22 – 28, 2023		 Print		Week	Month	Agenda				
Sun 1/22		Mon 1/23		Tue 1/24		Wed 1/25		Thu 1/26		Fri 1/27		Sat 1/28	
5am													
6am													
7am													
8am													
9am													

# Syllabus

CSE 4/545

Syllabus

Piazza

Schedule

Homeworks ▾

Mini Project ▾

Autolab

Book

# CSE 445/545 (Coding Theory) Syllabus

Spring 2023

Mondays, Wednesdays and Fridays, 4:00-4:50pm, [Cooke](#) ↗ 121.

## Under Construction

This page is still under construction. In particular, nothing here is final while this sign still remains here.

## Please note

It is **your responsibility** to make sure you read and understand the contents of this syllabus. If you have any questions, please contact the instructor.

## Academic Integrity



# Schedule

[CSE 4/545](#)[Syllabus](#)[Piazza](#)[Schedule](#)[Homeworks ▾](#)[Mini Project ▾](#)[Autolab](#)[Book](#)

## CSE 445/545 Spring 23 Schedule

Previous schedule: [2013](#), [2019](#), [2022](#).

### Under Construction

This page is still under construction. In particular, nothing here is final while this sign still remains here.

### Future Lectures

The topics for lectures in the future are tentative and subject to change.

**Date**

**Topic**

**Notes**

Mon, Jan 30

Introduction  [S22](#)

Wed, Feb 1

Definitions-I  [S22](#)

[Book: Sec 1.1, 1.2 and 1.3]

Fri, Feb 3

Definitions-II  [S22](#)

[Book: Sec 1.3 and 1.4]

# Autolab

[CSE 4/545](#)[Syllabus](#)[Piazza](#)[Schedule](#)[Homeworks ▾](#)[Mini Project ▾](#)[Autolab](#)[Book](#)

## Autolab

Details on Autolab, which will be used for all homework submissions in CSE 4/545.

### Under Construction


This page is still under construction. In particular, nothing here is final while this sign still remains here.

### The main link

We will be using the UB CSE extension to [Autolab](#)  for submission and grading of CSE 4/545 homeworks. You can access Autolab via <https://autograder.cse.buffalo.edu/> .

## Signing up

Follow these steps to setup an account on Autolab (unless you already have one in which case you'll use your existing account):

1. Go to [this page and click on the Sign in with MyUB link](#) . A new account will automatically be created for you.
2. By default, Autolab will use your official UB first and last name. **If you have a different preferred name, please let us know ASAP.**
3. When you login, the system will ask you to put in your nickname. It seems like to use the system you have to put in a nickname (though it won't be used for anything in this course).
4. After you have done the above steps, you wait.

# Piazza

[CSE 4/545](#)[Syllabus](#)[Piazza](#)[Schedule](#)[Homeworks ▾](#)[Mini Project ▾](#)[Autolab](#)[Book](#)

# CSE 445/545: Coding Theory

## Spring 2023

### Under Construction





This page is still under construction. In particular, nothing here is final while this sign still remains here.

### CSE 4/545 events

Today	◀	▶	Jan 22 – 28, 2023 ▾					Print	Week	Month	Agenda ▾
		Sun 1/22	Mon 1/23	Tue 1/24	Wed 1/25	Thu 1/26	Fri 1/27	Sat 1/28			
5am											
6am											
7am											
8am											
9am											

# Piazza for discussion

If you signed up before on 9pm on Sun, Jan 22 you should be on it

 note @13   

stop following **27 views**

Actions ▾

## Welcome to Piazza!

Students,

Welcome to Piazza! We'll be conducting all class-related discussion here this term. The quicker you begin asking questions on Piazza (rather than via emails), the quicker you'll benefit from the collective knowledge of your classmates and instructors. We encourage you to ask questions when you're struggling to understand a concept—you can even do so anonymously.

-Atri Rudra

other

Edit

good note | 0

Updated 6 days ago by Atri Rudra

# Feedback polls already up

 note @6   

[stop following](#)

**26 views**

[Actions](#) ▾

## Background feedback

For me to get a better sense of your background, please fill in these piazza polls:

- Linear Algebra: [@7](#)
- Abstract Algebra: [@8](#)
- Probability: [@9](#)
- Algorithms: [@10](#)
- Complexity: [@11](#)
- Why are you taking this course?: [@12](#)

(I will pin this post so that it is visible.)

feedback

[Edit](#) [good note](#) | 1

Updated 6 days ago by Atri Rudra

# Questions/Comments?

If something doesn't work (e.g. you cannot post a comment),  
let me know

# References

Draft of a book I'm writing  
With Guruswami+Sudan

Standard coding theory texts  
MacWilliams and Sloane  
van Lint  
Blahut  
Handbook of coding theory

## Essential Coding Theory

[Venkatesan Guruswami](#), [Atri Rudra](#) and [Madhu Sudan](#)

If you have any comments, please email them to [atri@buffalo.edu](mailto:atri@buffalo.edu)

The plan is to put up a draft of the whole book sometime in 2022 (for real this time!).

### Current Version

Below is a PDF of the book with the chapters that are now stable.

**[Draft of the book](#)** (January 31, 2022)

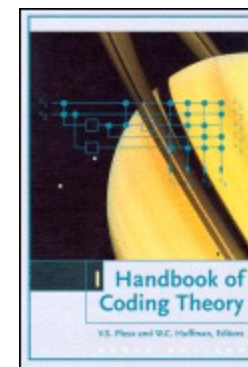
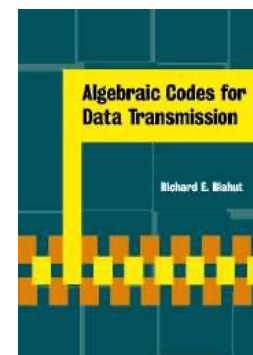
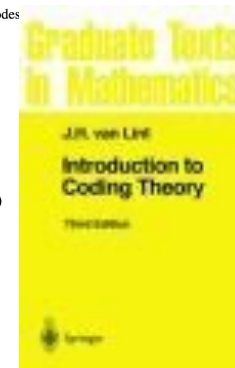
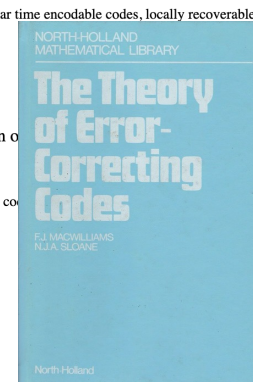
- (Major changes from last version: Added chapters on expander codes, linear time encodable codes, locally recoverable codes)

Warning: There are some dangling/missing links.

### Previous Versions

Listed below are previous versions of the book (in case you need an older version)

- [March 15, 2019](#).
  - (Major changes from last version: Added chapter on decoding RM codes)
- [December 18, 2018](#).
- [July 27, 2018](#).
- [Old version of the webpage that has separate chapter files](#).



# Pre-requisites

No formal pre-requisites for 545/ CSE 331 for 445

Probably no one will have all the pre-reqs

Mathematical maturity

Comfortable with proofs

Willing to pick up basics of new areas

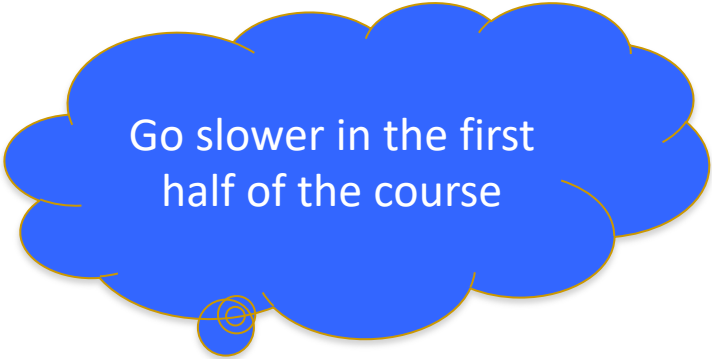
Will spend one lecture on the pre-req's

Linear Algebra

Finite Fields

Probability

Algorithms/ Asymptotic Analysis



Go slower in the first  
half of the course



# Grades and such like

## Grading Policy

Here is the split of grades:

Course Component	% of grade
Mini project	50%
Homeworks	50%

# Mini Project

Groups of size = 3

Create a Youtube video related to coding theory

Bunch of other details in syllabus

# Deadlines

**February 15, 2023.** You form groups of size **exactly three (3)** for the project by **11:59pm**. **One submission per group is needed to "register" the group.**

**March 1, 2023.** Your group submits the topic for your video by **11:59pm**. **One submission per group is needed to "register" the group.**

**March 29, 2023.** You should submit your two-page report by **11:59pm** on [Autolab](#).

**May 14, 2023.** You should submit your video by **11:59pm** on [Autolab](#).

# Deadlines are not suggestions

**February 15, 2023.** You form groups of size **exactly three (3)** for the project by **11:59pm**. **One submission per group is needed to "register" the group.**

## Zero if you miss this deadline

You will get a **ZERO** on the entire mini-project if you miss this deadline. So please make sure you submit the Google form (well) before the deadline.

**March 1, 2023.** Your group submits the topic for your video by **11:59pm**. **One submission per group is needed to "register" the group.**

## Zero if you miss this deadline

You will get a **ZERO** on the report and video parts of the mini-project if you miss this deadline. So please make sure you submit the Google form (well) before the deadline.

**March 29, 2023.** You should submit your two-page report by **11:59pm** on [Autolab](#).

## Zero on video if you miss this deadline

The entire group will get a **ZERO** on the video part of the mini-project if you miss this deadline. So please make sure you submit the report (well) before the deadline.

**May 14, 2023.** You should submit your video by **11:59pm** on [Autolab](#).

# Questions/Comments?

Check out the syllabus for more details

# Homework

6 short ones

Collaboration generally allowed

- Work in groups of size at most 3

- Write up your own solutions

- Acknowledge your collaborators

- No source other than book and your notes

- Breaking these rules will be considered as cheating

More details when they are handed out

# My homework philosophy for 4/545

**NOT** to make sure you understand what I teach in the lectures

Homework problems either

- Proofs that were not done in the class; or

- Material that is not covered in the class

- Closely related to something that is

The lectures will **NOT** “teach” you how to do your HWs

# This is a THEORY elective course

If you do not know how to write mathematical proofs on your own,  
you should DROP the course

HWs will need solid background in

Linear Algebra  
Probability

(CSE 545LEC) Teaching was fine. Grading and assignments are downright pathetic. Ive never seen a class where over half the class receives a zero in more than half the assignments and still the teacher never felt like there was a problem. Any decent professor would maybe decrease the difficulty of the assignments or try to gather from the students what is going wrong. Grading was pretty sub-par as well. Ive never in my entire student life where the median of an assignment is 0. Assignment after assignment a large majority of the class received a 0 and yet the professor did absolutely nothing. Never seen a professor so out of touch with his students.



# Questions/Comments?

Check out the syllabus for more details

# Accessibility Resources

## Information included in the syllabus

In short, let me know and consult with Accessibility Resources

# Preferred Name

If you prefer using name diff from UB records

Let me know and we'll make a note of it.

# Some comments

Decide on a Video topic **early**

Different topics might need different prep. work

Come talk to me

Homeworks might take time

Do not wait for the last moment

# Academic Dishonesty

All your submissions must be your own work

Penalty:

Minimum: An **grade reduction in course**

Possible: **F** (or higher penalty) if warranted

**YOUR** responsibility to know what is cheating, plagiarism etc.

If not sure, come talk to me

Excuses like “I have a job,” “This was OK earlier/in my country,” “This course is hard,” etc. **WON’ T WORK**

**I DO NOT HAVE ANY PATIENCE WITH ANY CHEATING :**  
**YOU WILL GET A GRADE REDUCTION IN THE COURSE**  
**FOR YOUR FIRST MISTAKE**

# If grades are all you care about

You'll be fine if

You do your assignments **honestly**

Make a reasonable attempt at them

# If you took CSE 331

I'm assuming you're in this class because you wanted to be here

Less scaffolding/support material

Am happy to give pointers but if you need to makeup on some background knowledge, I expect you to pick up the material on your own

# Questions/Comments?

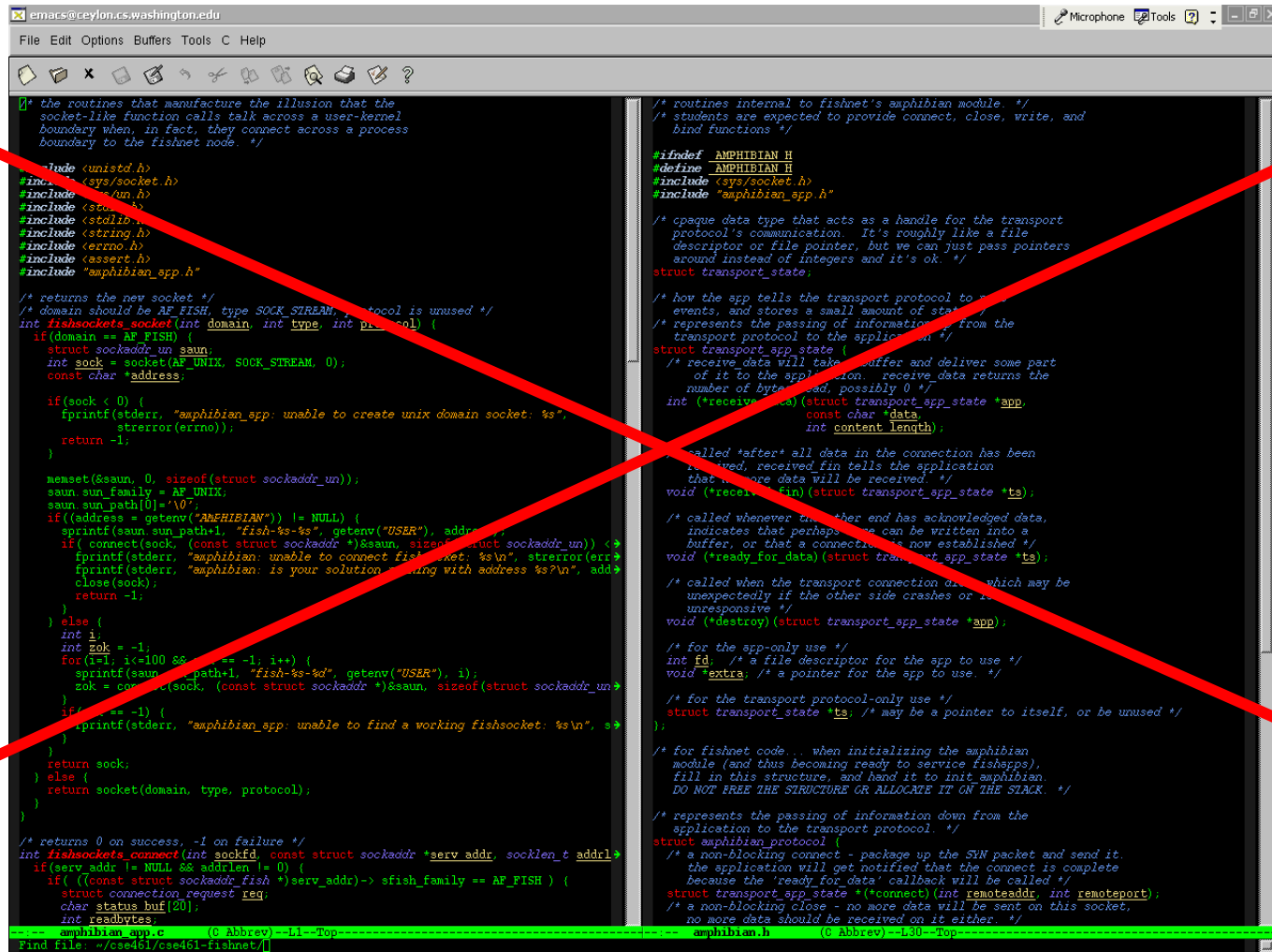
Check out the syllabus for more details



Let the fun begin!



# Coding theory



```
emacsc@ceylon.cs.washington.edu
File Edit Options Buffers Tools C Help

/* the routines that manufacture the illusion that the
socket-like function calls talk across a user-kernel
boundary when, in fact, they connect across a process
boundary to the fishnet node. */

#include <unistd.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <assert.h>
#include "amphibian_app.h"

/* returns the new socket */
/* domain should be AF_FISH, type SOCK_STREAM, protocol is unused */
int fishsockets_socket(int domain, int type, int protocol) {
    if (domain == AF_FISH) {
        struct sockaddr_un saun;
        int sock = socket(AF_UNIX, SOCK_STREAM, 0);
        const char *address;

        if (sock < 0) {
            fprintf(stderr, "amphibian_app: unable to create unix domain socket: %s",
                strerror(errno));
            return -1;
        }

        memset(&saun, 0, sizeof(struct sockaddr_un));
        saun.sun_family = AF_UNIX;
        saun.sun_path[0] = '\0';

        if ((address = getenv("AMPHIBIAN")) != NULL) {
            sprintf(saun.sun_path+1, "fish-%s-%s", getenv("USER"), address);
            if (!connect(sock, (const struct sockaddr *)&saun, sizeof(struct sockaddr_un))) {
                fprintf(stderr, "amphibian: unable to connect fishnet: %s\n", strerror(errno));
                fprintf(stderr, "amphibian: is your solution working with address %s?\n", address);
                close(sock);
                return -1;
            }
        } else {
            int i;
            int zok = -1;
            for (i=1; i<=100 && zok == -1; i++) {
                sprintf(saun.sun_path+1, "fish-%s-%d", getenv("USER"), i);
                zok = connect(sock, (const struct sockaddr *)&saun, sizeof(struct sockaddr_un));
            }
            if (zok == -1) {
                fprintf(stderr, "amphibian_app: unable to find a working fishsocket: %s\n", strerror(errno));
                return -1;
            }
        }
        return sock;
    } else {
        return socket(domain, type, protocol);
    }
}

/* returns 0 on success, -1 on failure */
int fishsockets_connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen) {
    if (serv_addr != NULL && addrlen != 0) {
        if (((const struct sockaddr_fish *)serv_addr)->sfish_family == AF_FISH) {
            struct connection_request req;
            char status buf[20];
            int readytimes;

            amphibian_app.c (0 Abbrev) --L1--Top-----
            Find file: ~/cse461/cse461-fishnet/

/* routines internal to fishnet's amphibian module. */
/* students are expected to provide connect, close, write, and
bind functions */

#ifndef AMPHIBIAN_H
#define AMPHIBIAN_H
#include <sys/socket.h>
#include "amphibian_app.h"

/* opaque data type that acts as a handle for the transport
protocol's communication. It's roughly like a file
descriptor or file pointer, but we can just pass pointers
around instead of integers and it's ok. */
struct transport_state;

/* how the app tells the transport protocol to do things,
events, and stores a small amount of state. */
/* represents the passing of information down from the
transport protocol to the application */
struct transport_app_state {
    /* receive data will take a buffer and deliver some part
of it to the application. receive_data returns the
number of bytes read, possibly 0 */
    int (*receive_data)(struct transport_app_state *app,
        const char *data,
        int content_length);

    /* called 'after' all data in the connection has been
received, received_fin tells the application
that no more data will be received. */
    void (*received_fin)(struct transport_app_state *ts);

    /* called whenever the other end has acknowledged data,
indicates that perhaps more can be written into a
buffer, or that a connection has now established */
    void (*ready_for_data)(struct transport_app_state *ts);

    /* called when the transport connection dies, which may be
unexpectedly if the other side crashes or is
unresponsive */
    void (*destroy)(struct transport_app_state *app);

    /* for the app-only use */
    int fd; /* a file descriptor for the app to use */
    void *extra; /* a pointer for the app to use. */

    /* for the transport protocol-only use */
    struct transport_state *ts; /* may be a pointer to itself, or be unused */
};

/* for fishnet code... when initializing the amphibian
module (and thus becoming ready to service fishapps),
fill in this structure, and hand it to init_amphibian.
DO NOT FREE THE STRUCTURE OR ALLOCATE IT ON THE STACK. */

/* represents the passing of information down from the
application to the transport protocol. */
struct amphibian_protocol {
    /* a non-blocking connect - package up the SYN packet and send it.
the application will get notified that the connect is complete
because the 'ready_for_data' callback will be called */
    struct transport_app_state *(*connect)(int remoteaddr, int remoteport);
    /* a non-blocking close - no more data will be sent on this socket,
no more data should be received on it either */
    void (*close)(int remoteaddr);
};
#endif
```

What does this say?

W\*Icome to the cl\*ss. I h\*pe you w\*ll h\*ve as mu\*h f\*n as I  
wi\*I hav\* t\*ach\*ng it!

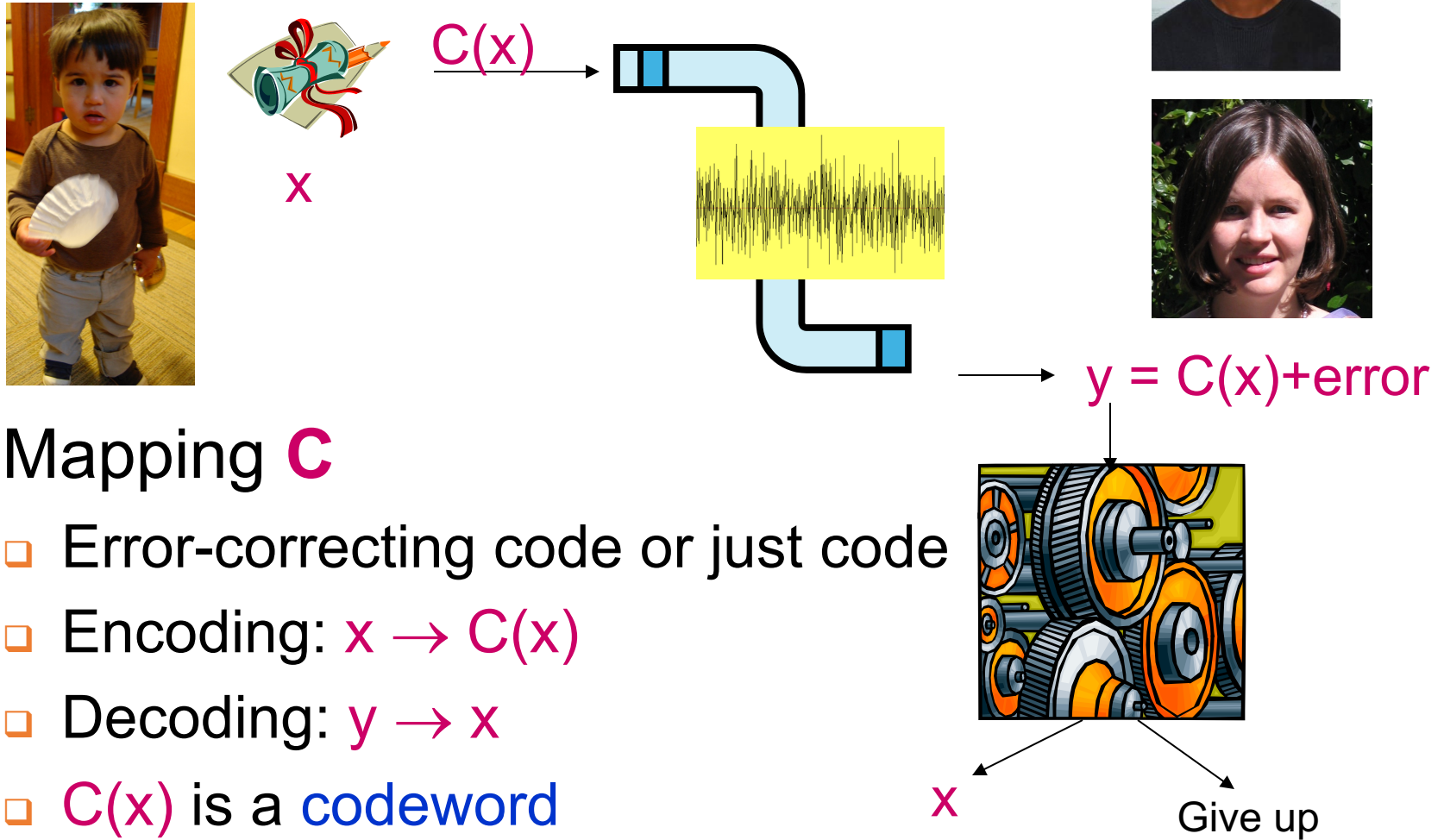
Welcome to the class. I hope you will have as much fun as I  
will have teaching it!

# Why did the example work?

English has in built redundancy

Can tolerate “errors”

# The setup



# Communication

## Internet

Checksum used in multiple layers of TCP/IP stack

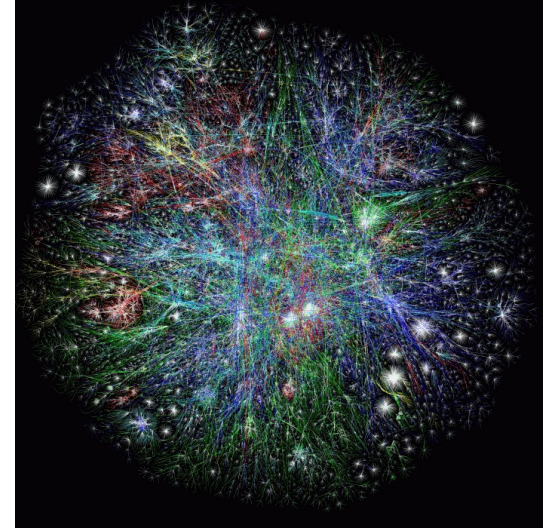
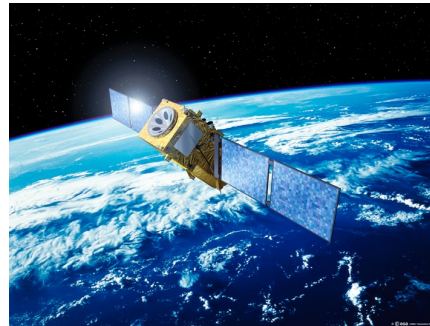
## Cell phones

## Satellite broadcast

TV

## Deep space telecommunications

Mars Rover





# Codes and 5G

UC San Diego News Center

October 11, 2018 | By Daniel Kane

## Samsung Licenses 5G Polar Coding Technology Developed by UC San Diego Engineers

Samsung and the University of California San Diego recently signed a major license agreement for the telecommunications industry, for a standard-essential error-correction technology developed by engineers from the Jacobs School of Engineering.

This new technology enables a new



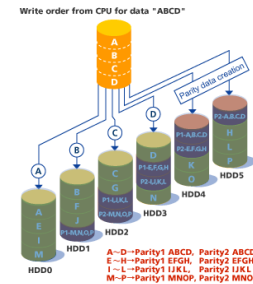
# “Unusual” applications

## Data Storage

CDs and DVDs

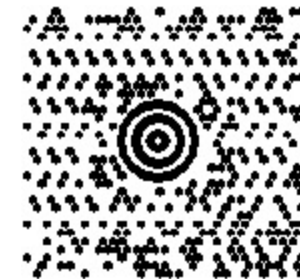
RAID

ECC memory



## Paper bar codes

UPS (MaxiCode)



Codes are all around us



# While applications are numerous...

This course (lectures and HWs) will focus ONLY on proofs

where in the above “ $|E(\mathbf{m})$ ” is short for “being conditioned on  $E(\mathbf{m})$  being transmitted” and the inequality follows from the union bound (Proposition 3.1.5) and the fact that  $D$  is MLD.

Noting that  $\Delta(E(\mathbf{m}'), \mathbf{y}) \leq \Delta(E(\mathbf{m}), \mathbf{y}) \leq (p + \epsilon')n$  (see Figure 6.6), by (6.9) we have

$$\begin{aligned} \mathbb{E}_E [\mathbb{1}_{D(\mathbf{y}) \neq \mathbf{m}}] &\leq \sum_{\mathbf{m}' \neq \mathbf{m}} \Pr [E(\mathbf{m}') \in B(\mathbf{y}, (p + \epsilon')n) | E(\mathbf{m})] \\ &= \sum_{\mathbf{m}' \neq \mathbf{m}} \frac{|B(\mathbf{y}, (p + \epsilon')n)|}{2^n} \end{aligned} \quad (6.10)$$

$$\begin{aligned} &\leq \sum_{\mathbf{m}' \neq \mathbf{m}} \frac{2^{H(p + \epsilon')n}}{2^n} \\ &< 2^k \cdot 2^{-n(1 - H(p + \epsilon'))} \end{aligned} \quad (6.11)$$

$$\leq 2^{n(1 - H(p + \epsilon)) - n(1 - H(p + \epsilon'))} \quad (6.12)$$

$$= 2^{-n(H(p + \epsilon) - H(p + \epsilon'))}. \quad (6.13)$$

In the above, (6.10) follows from the fact that the choice for  $E(\mathbf{m}')$  is independent of  $E(\mathbf{m})$ . (6.11) follows from the upper bound on the volume of a Hamming ball (Proposition 3.3.3), while (6.12) follows from our choice of  $k$ .

Using (6.13) in (6.8), we get

$$\begin{aligned} \mathbb{E}_E \left[ \Pr_{\mathbf{e} \sim \text{BSC}_p} [D(E(\mathbf{m}) + \mathbf{e}) \neq \mathbf{m}] \right] &\leq e^{-(\epsilon')^2 n/2} + 2^{-n(H(p + \epsilon) - H(p + \epsilon'))} \sum_{\mathbf{y} \in B(E(\mathbf{m}), (p + \epsilon')n)} \Pr [\mathbf{y} | E(\mathbf{m})] \\ &\leq e^{-(\epsilon')^2 n/2} + 2^{-n(H(p + \epsilon) - H(p + \epsilon'))} \leq 2^{-\delta' n}, \end{aligned} \quad (6.14)$$

where the second inequality follows from the fact that

$$\sum_{\mathbf{y} \in B(E(\mathbf{m}), (p + \epsilon')n)} \Pr [\mathbf{y} | E(\mathbf{m})] \leq \sum_{\mathbf{y} \in \{0,1\}^n} \Pr [\mathbf{y} | E(\mathbf{m})] = 1$$

and the last inequality follows for large enough  $n$ , say  $\epsilon' = \epsilon/2$  and by picking  $\delta' > 0$  to be small enough. (See Exercise 6.3.)

Thus, we have shown that for any arbitrary  $\mathbf{m}$  the average (over the choices of  $E$ ) decoding error probability is small. However, we still need to show that the decoding error probability is exponentially small for *all* messages *simultaneously*. Towards this end, as the bound holds for each  $\mathbf{m}$ , we have

$$\mathbb{E}_{\mathbf{m}} \left[ \mathbb{E}_E \left[ \Pr_{\mathbf{e} \sim \text{BSC}_p} [D(E(\mathbf{m}) + \mathbf{e}) \neq \mathbf{m}] \right] \right] \leq 2^{-\delta' n}.$$

# Other applications of codes

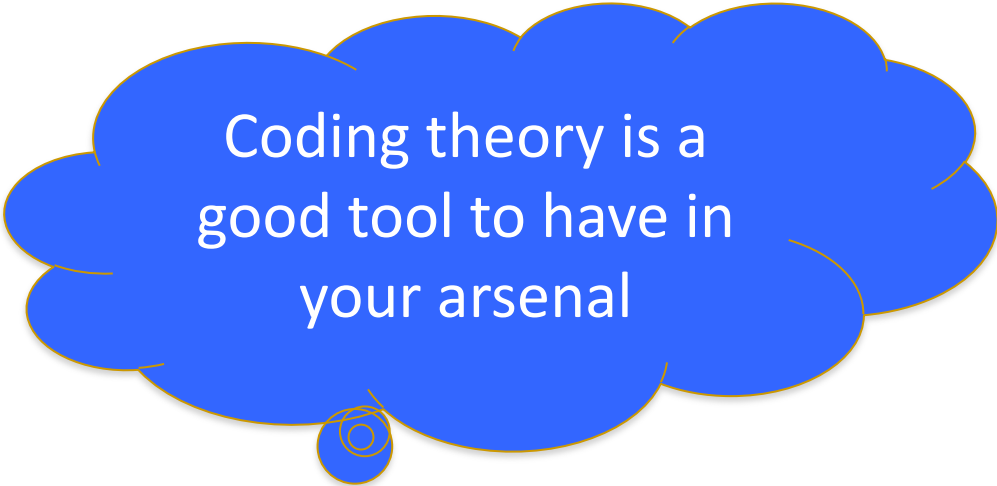
Outside communication/storage domain

Tons of applications in theory

- Complexity Theory

- Cryptography

- Algorithms



Coding theory is a  
good tool to have in  
your arsenal