Lecture #1: Overview & Design Principles #1

WPI CS4516 Spring 2019 D term

Instructor: Lorenzo De Carli (ldecarli@wpi.edu)

(slides include material from Christos Papadopoulos, CSU)

Course topics

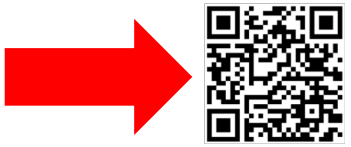
- Review of **foundational topic** in computer networks
- Based on **reading list** of **scientific papers** from **networking literature**
  - You are responsible for reading each paper **before lecture**
- Focus on works with **long-lasting**, **practical impact** on networking
  - If interested in **cutting-edge research topics** and/or **theoretical work** look at **graduate-level offerings**

Course organization

- Every week we will cover a relevant networking topic
  - Internet design principles
  - Network layer
  - Transport layer
  - Network measurements and monitoring
  - Network management (2 weeks)
  - Router design
  - Net neutrality
- During every lecture we will discuss an **influential paper** related to that week's topic

Course material

- **Reading list** of 13 papers
  - **No textbook**
  - I may provide **optional material** to aid understanding
- **Lecture slides**
- You are expected to **read each paper before coming to lecture**
- **Discussion** with your classmates is **encouraged**
- I will set up **forums on Canvas** that you can use to **engage in conversation** with your classmates and the instructor



Resources & course management

- Course overview, syllabus, assigned readings on class website:
  - https://web.cs.wpi.edu/~ldecarli/teaching/cs4516D19
- Same material also available on Canvas
  - https://canvas.wpi.edu/courses/12976
- Canvas will be used for uploading assignments, taking quizzes, interacting w/ students & instructor, posting grades

Class project

- The class also requires to complete a **practical project**
- Project goal: implement a gateway to perform **application identification** from **encrypted traffic**
- **4 project deliverables** to be completed at different times during the term
- The project will be introduced in the next lecture

Grading

- Six 20-minute **in-class quizzes**: 60% of grade
  - Each quiz at the **beginning of each week**, except for week #1 (**no quiz**) and week #7 (**quiz at the end of the week**)
  - Each quiz covers the **material from the previous week** (material discussed in class **and papers**)
  - Just to be clear: **content of the papers** (even if not discussed in class) **is fair game for quizzes**
  - I will **drop the lowest quiz score** when computing the final grade
    - This also mean **you can miss one quiz w/o impacting your grades**

Grading - II

- Class project: 40% of grade
  - Project will consist of **four phases**
  - At the end of phases 1-3, you will need to deliver **what you implemented so far** (each phase builds on the previous one)
  - At the end of phase 4, you will need to deliver a **final report and the project implementation**
  - Each phase will receive a separate grade. The overall quality of the implementation will also be evaluated.

Participation, behavior etc.

- **“Good behavior coefficient”** (GBC)
- Final grade:
  - (0.6 * <quiz score> + 0.4 * <project score>) * **GBC**
- Takes into account **attendance** and **respectful behavior towards instructor, TAs, classmates** (or lack thereof)
- Can also go **> 1.0** ☺
- (If you receive a GBC != 1, I will tell you why)
- (Will also try to warn you, as reasonable, if **your behavior is at risk of affecting your grade**)

Lastly, who am I?

- **Lorenzo De Carli**
  - **Assistant Professor of Computer Science**
  - Politecnico di Torino (Torino, Italy):
    - B.Sc. (2005), M.Sc. (2007) in Computer Engineering
  - University of Wisconsin-Madison:
    - M.Sc. (2010), Ph.D. (2016) In Computer Science
  - Assistant Professor of CS at Colorado State University from 2017 to 2018
  - Now at WPI ☺
  - Interested in **Networking**, **Programming Languages**, **Security**

One more thing before we begin…

How to read papers

- In order to successfully understand scientific papers, it is important to learn **how** to read them

- **Common mistake:** read a paper sequentially the way you would read a book, dedicating equal attention to all parts
  - **It almost never works** - you will drown in the details, miss the big picture, and get very frustrated
  - Instead, try reading a paper selectively, **focusing on the most important parts first**

How to read papers - II

- The trick is to read a paper **multiple times**:
  - **First step:** read abstract, intro and conclusion; skim the beginning of every section in between. **Goal:** understand which problem the paper aims to solve, and get a very basic understanding of how it gets solved
  - **Second step:** read the paper end-to-end, trying to understand the technical details of how problems are solved.
  - For particularly complex papers, **you may need to repeat the second step multiple times**, every time delving deeper into the technical details

- **Additional readings** (linked on the course web page):
  - M. Hanson, “Efficient Reading of Papers in Science and Technology”
  - S. Keshav, “How to Read a Paper”

How to read papers - III

- **Summarizing:** skim the paper first to understand what is the problem being solved, and then try to refine your understanding with further readings

- What happens if you follow this method:
  - **Pass 1:** *this paper proposes some hardware which can execute different networking algorithms*
  - **Pass 2:** *this paper propose a dataflow-based processor which can efficiently run multiple forwarding and classification algorithms, overcoming the limitations of traditional network ASICs*

How to read papers - IV

- What happens if you don’t follow this method and try to understand everything at once:
  - Pass 1: this paper solves a problem in networking
  - Pass 2: this paper solves a problem in networking
  - Pass 3: this paper solves a problem in networking
  - Pass 4: this paper solves a problem
  - Pass 5: this paper
  - Pass 6: ZZzzzzz…

How to read papers - V

- Other tips:
  - **Take notes**
  - Form informal **reading groups** with your classmates so you can read and discuss the paper together
  - **Come to the lecture prepared for discussion** (especially if there are aspects of a paper which you did not understand)

Now, let’s get down to business…

Internet design #1: how to design complex systems

Reading: "End-to-end Arguments in System Design"

J.H. Saltzer, D.P Reed & D.D. Clark

Let’s begin from the beginning

- Systems of networked computers (e.g., the Internet) are **incredibly complex objects**
  - **Too complex for us humans** to fully grasp!
  - So how do us, puny humans, go about building one?
- We need a plan!
- … or more precisely, some **guiding principles**…
- …on which **everyone** agrees

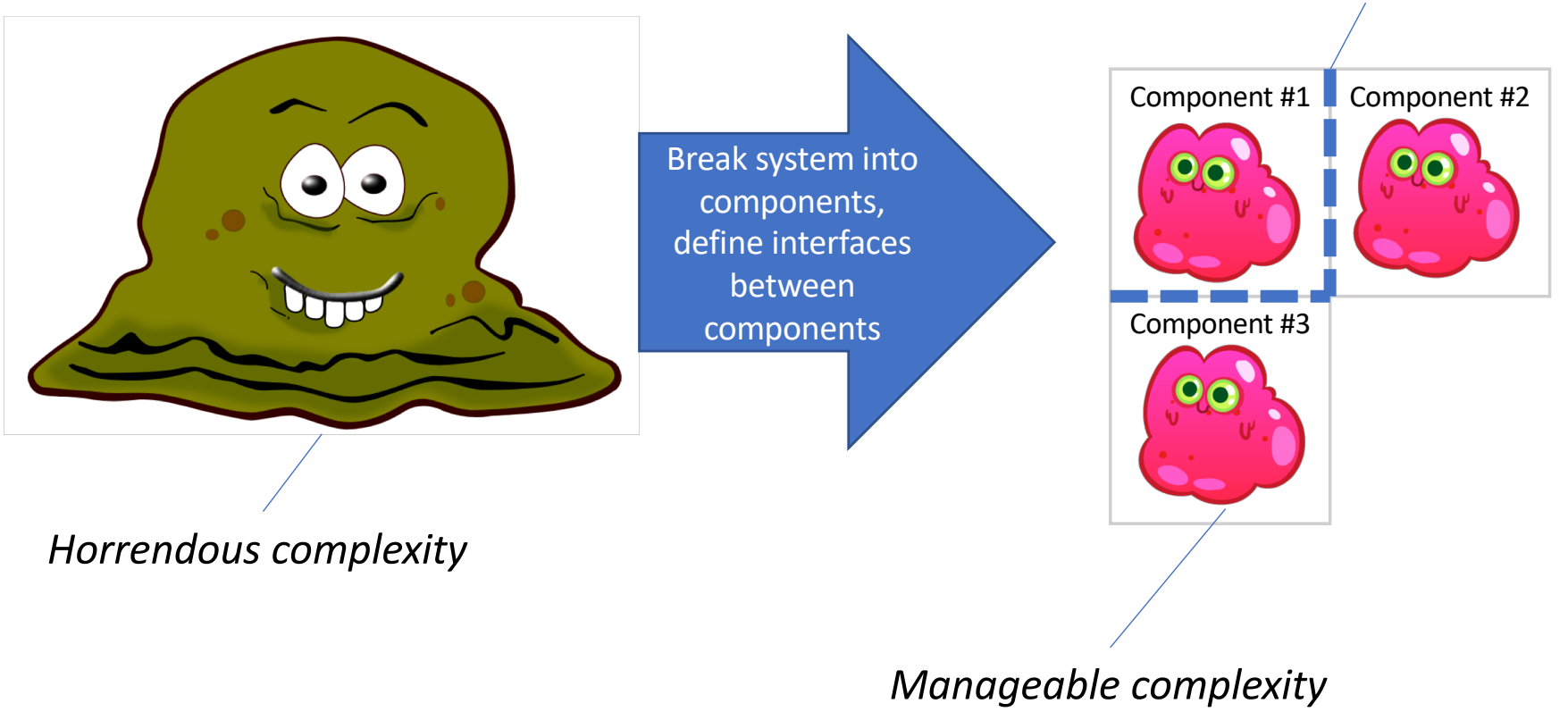
A general principle

- "**Modularity** guided by **abstraction** is the way **things get done**" (Scott Shenker)
  - Can you explain me what this means?

- Pretty much **every idea that everyone has ever had** in the field of designing computer systems implements this principle!

So, this is what we need to do…

Suppose that we need to design a system with **horrendous complexity…**

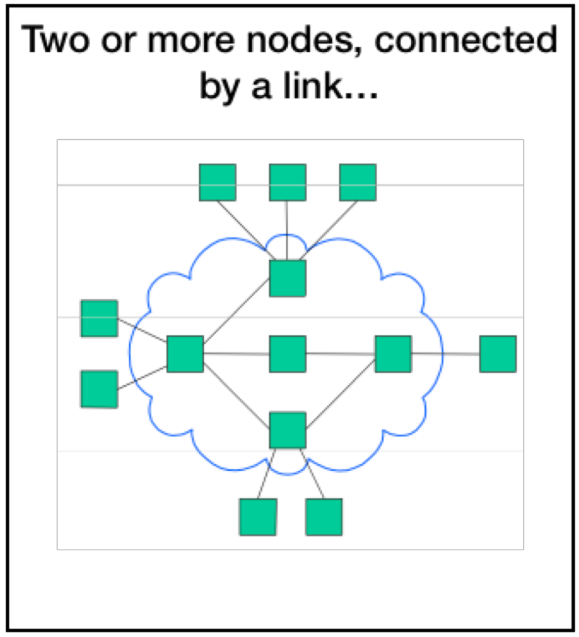


This is still too general

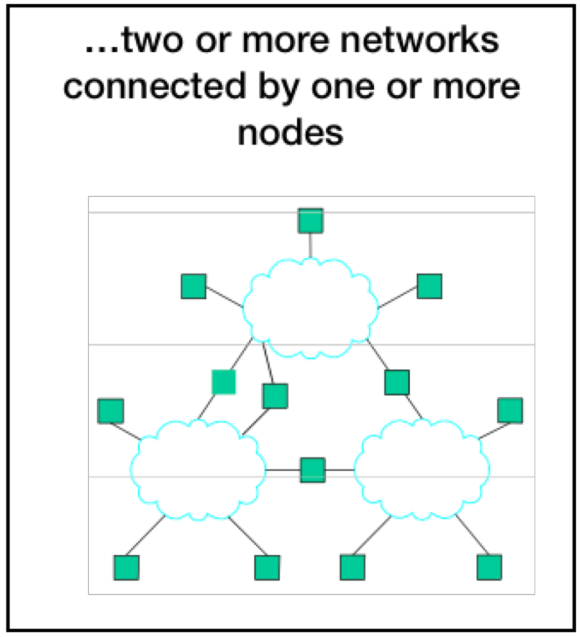
- The idea of breaking a complex system into simpler components is powerful…
- …but still too general!
- How to translate it to concrete design principles?
  - Particularly in the domain of **designing a network**?

First things first

- What is a network?
  - A computer network can be recursively defined as:



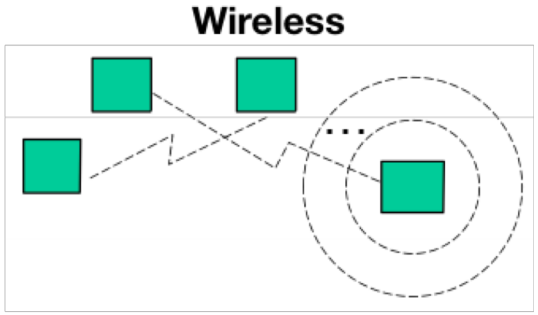
…OR…



Nodes and links

- A **node** is pretty much any computing device that can communicate with other devices (PC, phone, smart device, car, etc.)
- A **link** is a component (physical or logical) which two or more nodes use to exchange data
- **Examples of links:**

x



Where is the complexity in a networked system?

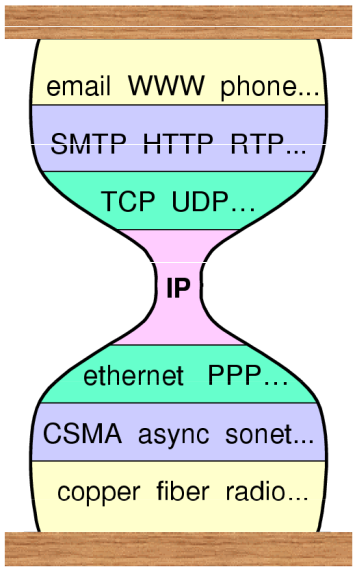
- Any guesses?
- Some core problems:
  - Making effective use of the physical medium (relatively easy w/ wires, less so with wireless)
  - Figuring out how to reach a specific destination
  - Ensuring communication can continue even if individual links fail
  - Multiplex many communication flows on the same links
  - Allowing different applications to communicate using the same network infrastructure
  - … (more on this in lecture #2)

How do networks break this complexity?

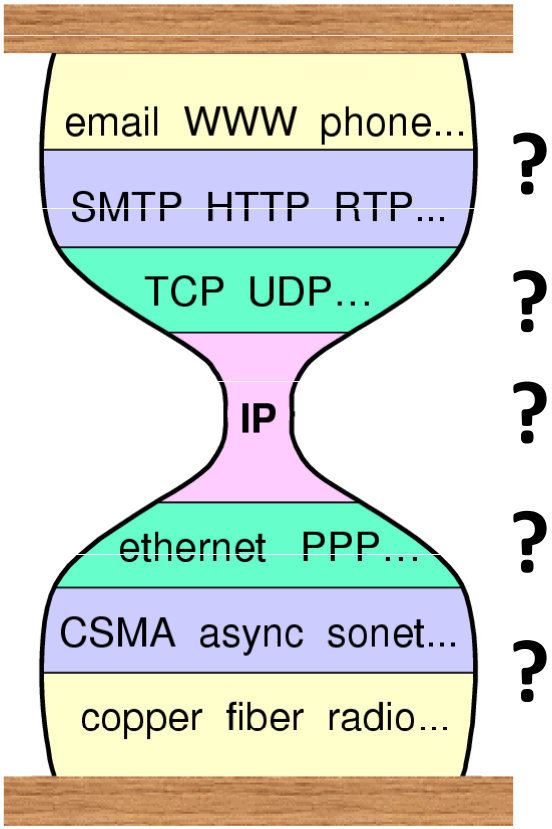
1. Identify a **set of core tasks** that the network infrastructure must perform
2. Define **how** the infrastructure must perform each task
    1. These definitions are usually called **protocols**
3. Define how protocols **interact** with each other

Protocol stack

- Computer networks are based on a **stack of protocols** which provide features at **increasingly high level of abstraction**
- The **hourglass** design:



Review time!



Application (high-level data stream)

Transport (reliable transmission of packets)

Network (routing across multiple networks)

Data-link (transmission of frames on link)

Physical (characteristics of physical signal)

Protocol stack - II

- Today we won't discuss in the details the functions of each layer…

- Instead, we will review a **general principle** that helps designers to decide **how to distribute functions across layers**

End-to-end arguments in system design

- 1981 paper by Jerry Saltzer, David Reed, Dave Clark
  - MIT Laboratory for Computer Science
  - All authors went on to make important contribution to the Internet design and architecture

- **Heuristic** for solving the problem of **function placement** among the modules of a distributed system

- Highly influential

The end-to-end argument

- Architectural principle

- Deals with where to place protocol functionality
  - Inside the network (i.e. in switches/routers), or
  - At the edges (endpoints)

The core principle

If a function can only be fully and correctly implemented with the help of the applications standing at the endpoints of a communication system, providing that function as a feature of the communication system is not advisable

Can you explain what it means?

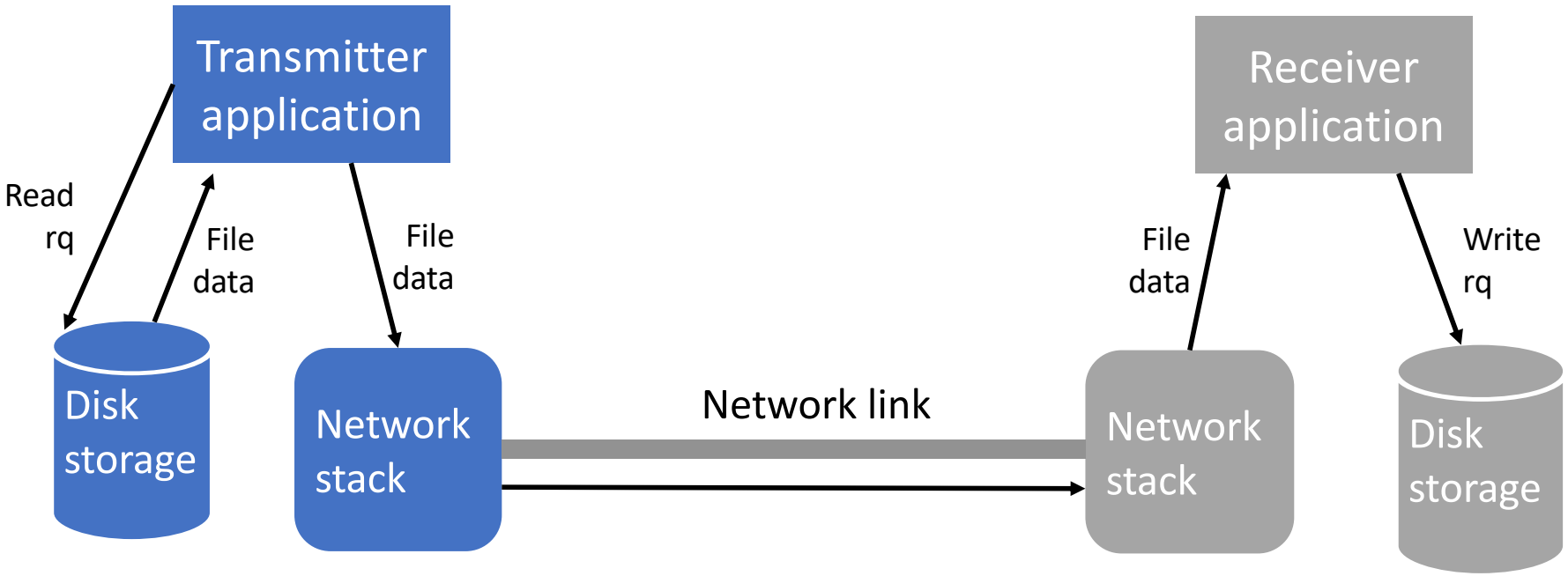
Motivation

- Don't duplicate functionality at multiple levels if you have to do it at the top anyway

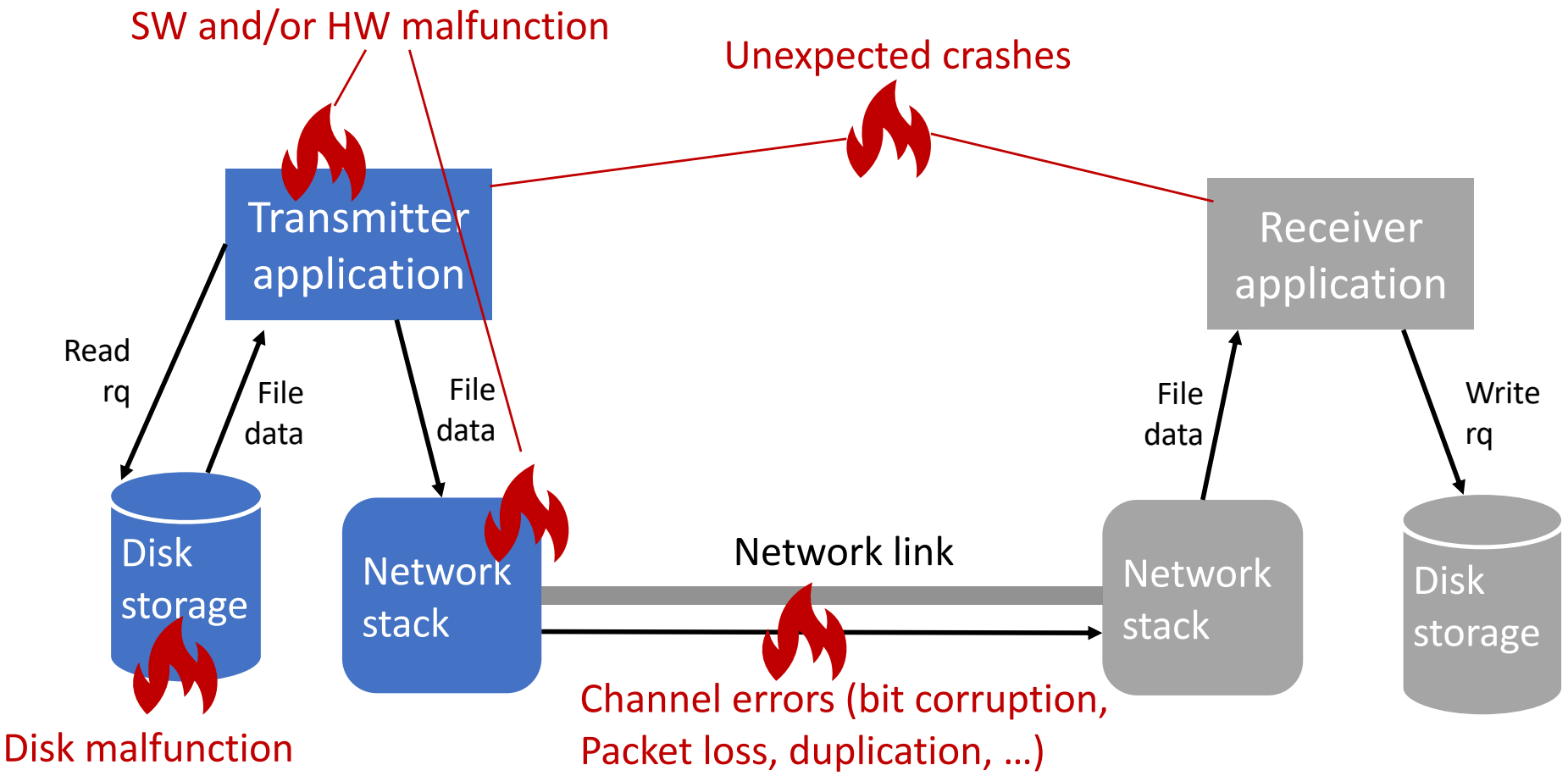
- Why? Two main reasons:
  - Duplicate functionality has a cost associate with it (e.g. reliable transport introduces delays and retransmissions)
  - Functionality provided at low level is forced upon all applications, even those which do not need it or would work better without it

Example

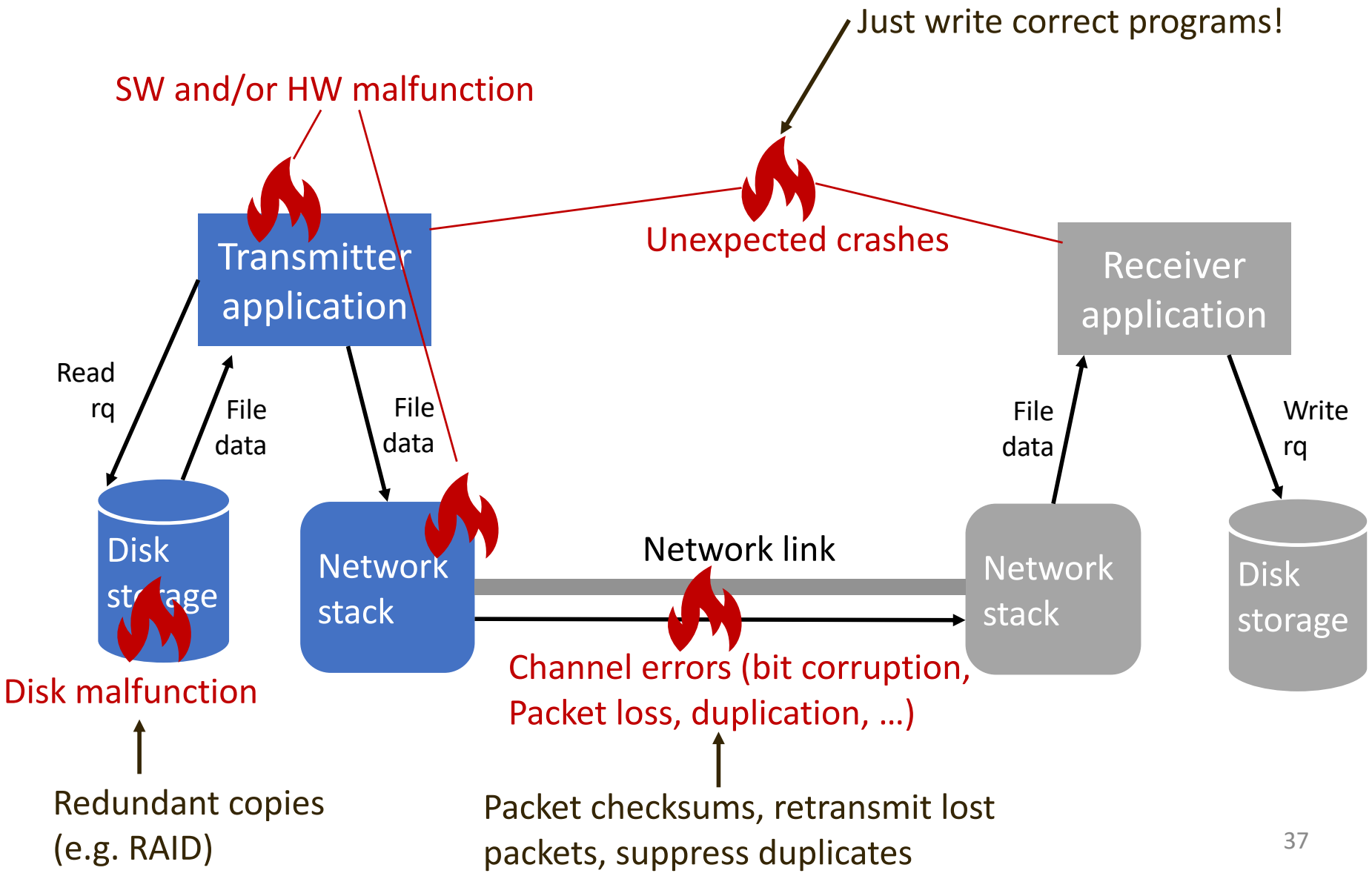
- **File transfer application**
- Components:



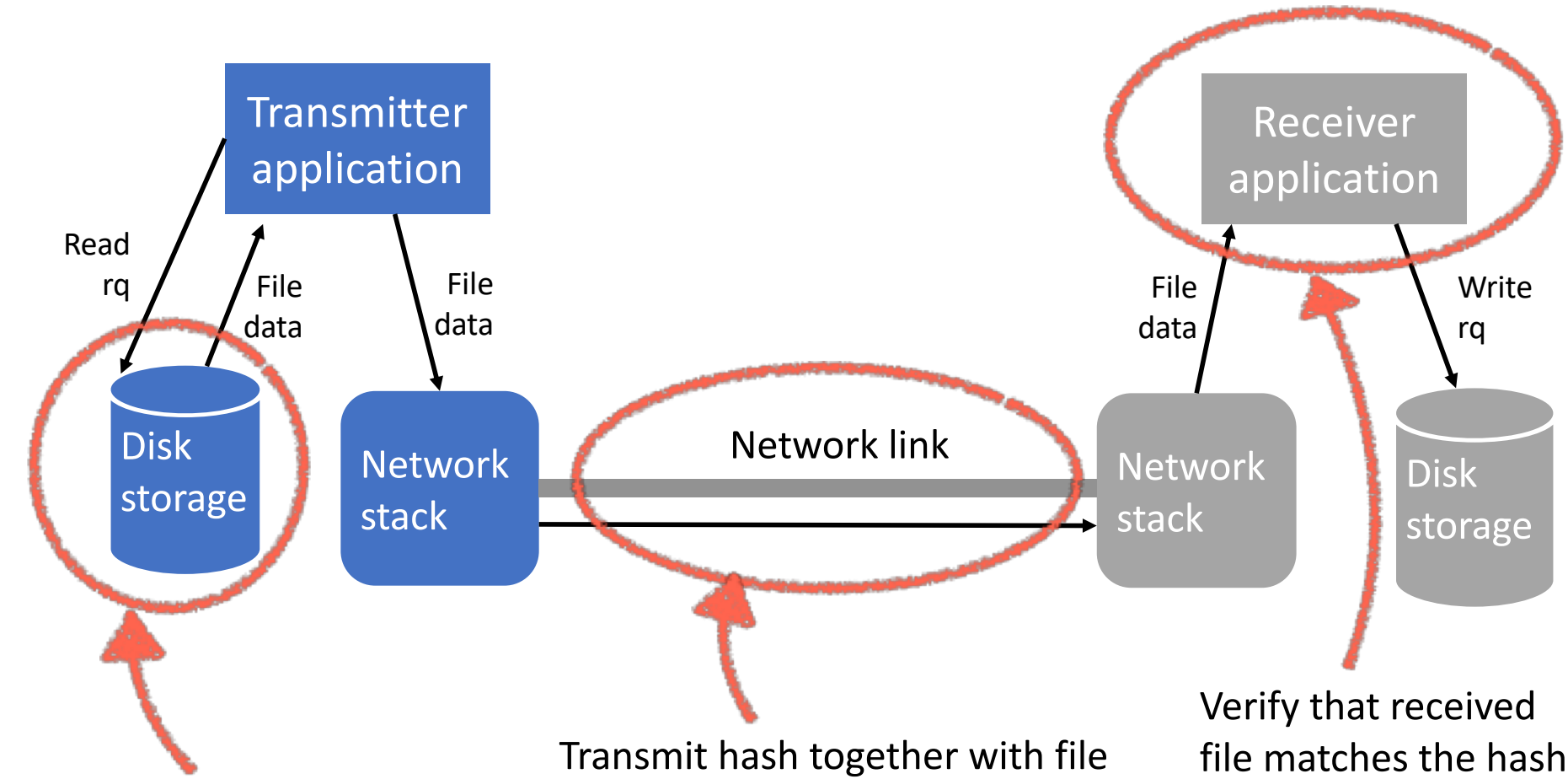
What could possibly go wrong?



How would you go about solving the problem?



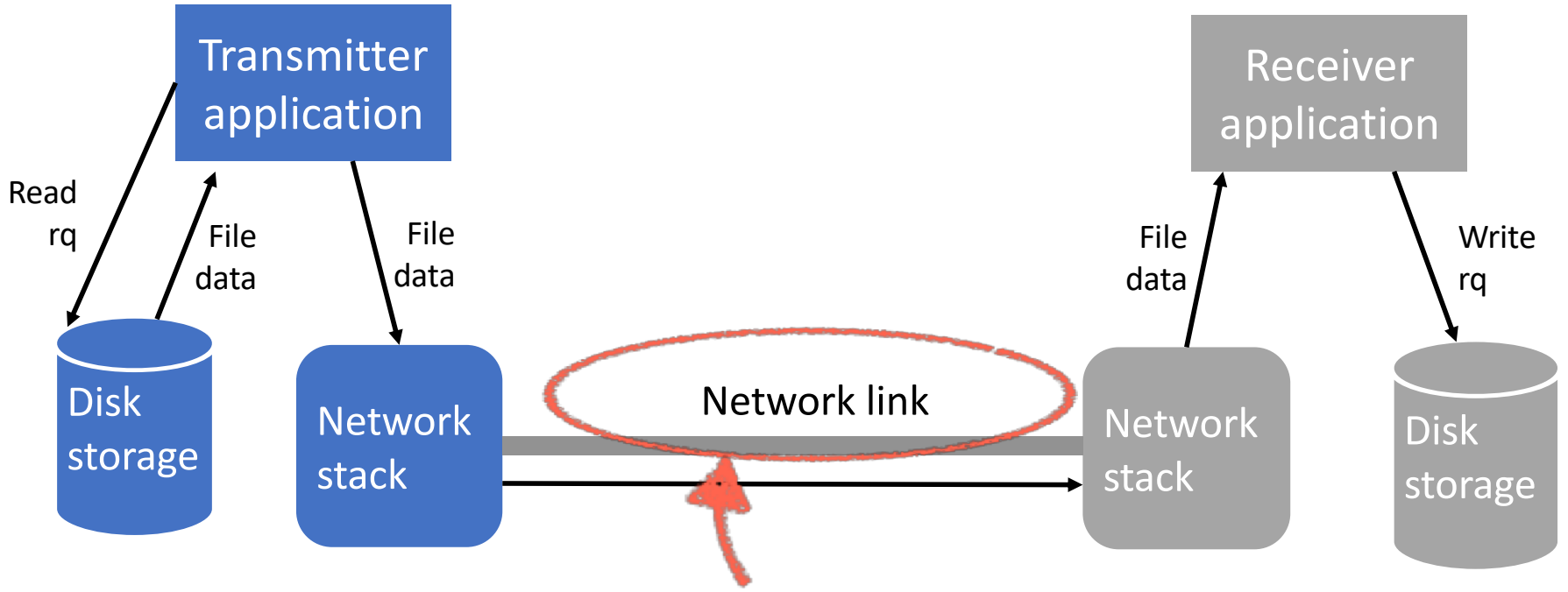
You could do all of this… or …



Why is this a good idea?

- Guards against all malfunctions we have considered: errors are discovered regardless if they originate in faulty disks, SW/HW or network links
- Simple and economical: requires no built-in redundancy and error correction in the system
- If everything works correctly, most transfers will be completed at first attempt
  - If a transfer require more than a couple of tries to be completed, there is probably an issue in the system

Alternative approach: reliable network transmission



Ensure transmission errors are detected and corrected

Does not solve the problem: I still need to transmit and check the file hash to guard against all other possible errors

Another drawback of not respecting e2e

- False sense of security
- Why?
- MIT code repository example:

- Result: OS source gradually corrupted!

Everything is relative

- Correctness checks in individual components do not solve the reliability issue…
  - But they may make sense as optimizations!
- Consider a file which is transmitted over **100** packets. Probability of packet error is $\mathbf{p_E}$. What is the expected value **E[T]** of the number of trials **T** before the file is transmitted successfully, for various values of $\mathbf{p_E}$?

p	E[T]	
0.00001	~1	} Acceptable
0.0001	~1	
0.001	1.10	
0.01	2.73	} Maybe acceptable?
0.1	37648	} Definitely not acceptable!

What's the lesson here?

- Violations of e2e, while not recommended in general, may make sense as **optimizations**
- In this example, a highly unreliable network may reduce the chance of successful transmission to negligible values
- It makes sense to introduce reliability as a network features (e.g. automatically retransmit corrupted packets)
- When is this acceptable? As always in engineering, the answer is "it depends" :-)

Some case studies

Delivery guarantees

- Should the network acknowledge message reception?
  - **Argument in favor:** increases reliability, message integrity etc.
  - **Argument against:** increases delay - network stack must wait for acknowledgement before further transmission
    - What if app does not need it?
    - Can you think of an example?

Delivery guarantees - II

- The argument against delivery guarantees in the paper is that the sender really cares about the **effect** of the packet, not its **reception**
  - Distributed protocols, databases, etc. all incorporate some sort of explicit application-level commit - having a network-level acknowledgement is not enough!

Delivery guarantees - III

- The argument against delivery guarantees in the paper is that the sender really cares about the **effect** of the packet, not its **reception**

- **Can you think why this consideration may be outdated in some cases?**

Secure transmission of data

- Can you name a technology for true end-to-end data encryption?
  - IPsec - end-to-end-ish: encryption between IP endpoints, but not between applications
  - TLS?

Secure transmission - II

- Why is it a good idea to have e2e encryption?

- Why is it a good idea to have non-e2e encryption?

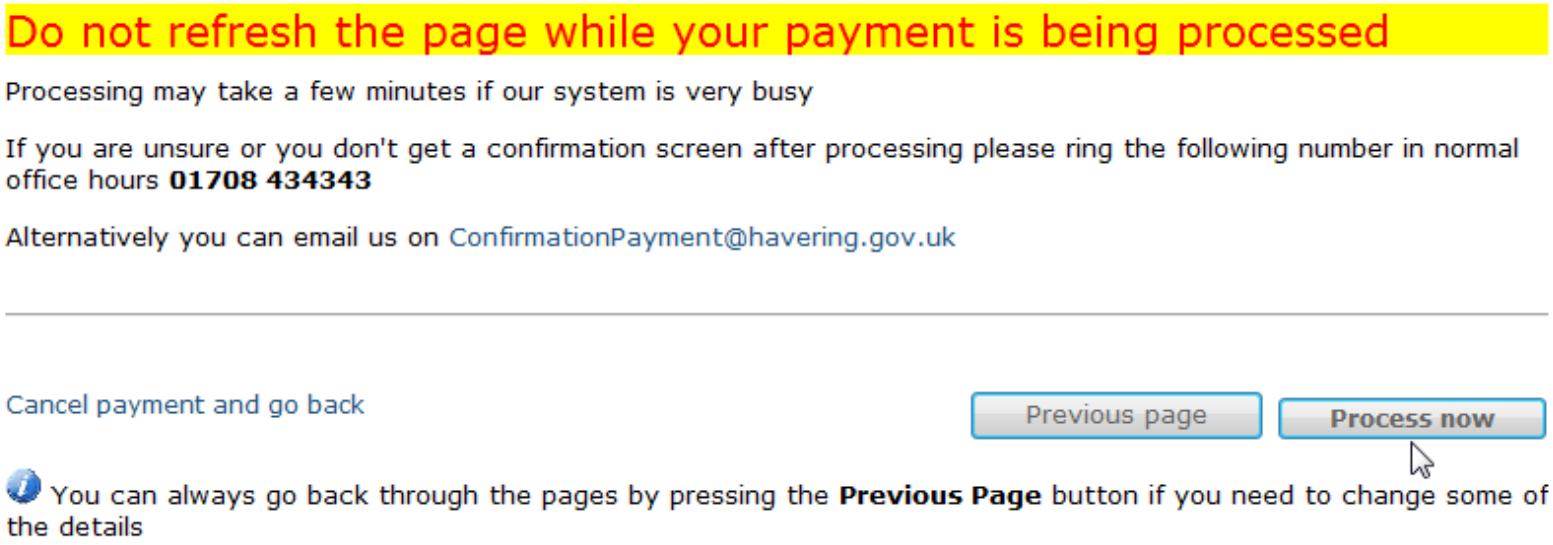
Guarantees encryption of all traffic!

Ordered message delivery

- May be difficult or impossible to provide within the network
  - E.g. packets may take different paths in the network due to transient routing failures
  - In that case, packets may arrive out of order - and there is little any entity participating in the communication can do

Duplicate message suppression

- Can you think of a reason why doing this in the network is not enough?



Nuances

- E2e specifies that functionality should be implemented at the endpoints - but it does not specified what those endpoints are
- In certain cases, the choice of endpoints is non-obvious and has important consequences on the system design

What is an endpoint?

- Example: voice communication

- Where should error correction go?

What is an endpoint? - II

- This use case does not imply that you should always let the user take care of the problem :-)
- Consider the case of a voicemail server which streams pre-recorded voice messages
  - Error correction is probably a good idea there
- **Lesson:** e2e cannot be applied blindly - designers must use some care in choosing the correct endpoints!

Not just for networking!

- E2e is often presented as a principle for distributed system design
- Which is interesting, because the paper presents several examples not related to distributed systems
- Do you remember which ones?

CPU design

- The old CISC vs RISC debate
  - CISC: complex functionality provided as processor instructions
  - RISC: processor only provides basic operation (conditional, arithmetic, etc.) - everything else build in software

OS design

- Monolithic OS vs microkernel
  - Monolithic: OS implements an extensive amount of functionality (networking, file system, etc.) in kernel space
  - Microkernel: OS provides isolation and interprocess communication, everything else is implemented outside the kernel (and does not affect its stability)
- Which one best implements the e2e idea?

Can we reformulate e2e in a non-networking specific way?

- *The correct level of abstraction at which to provide a function in a complex system is the level at which the most information about the function's purpose is available*
  - **CPU design:** complex instructions should be implemented in software, not in hardware
  - **OS design:** all primitives but process isolation and IPC should be implemented by user space libraries, not the kernel
  - **Compiler design:** program optimization should be performed by the compiler, not the programmer ("premature optimization is the root of all evil", Knuth)
  - **Network design:** high-level functionality should be implemented at the endpoints, not in the network core

A final consideration

- E2e is not a law - it is a **heuristic**
  - ***Heuristic:*** *a technique serving as an aid to problem solving by experimental and especially trial-and-error method* (Merriam-Webster dictionary)
- The paper we just discussed condenses lessons from years of experience in system design - but by no means this mean that this is always the correct approach
- E2e fosters **simplicity** and **elegance** of design, which are good things…
- But practical systems are often neither elegant not simple!

Occam's razor

- Heuristic principle to help in scientific investigation (and any investigation, really)
- Often misquoted as "the simplest solution is most likely correct"
- It should actually be interpreted as "the simplest solution **among all the plausible ones** is most likely correct"
- Keep in mind that **correctness > simplicity**

Breaking e2e

- Can you think of networking technologies that break e2e and yet are widely deployed?
  - NAT
  - Network monitoring

Coming soon

- Internet design principles
- Class project overview