Software Systems

Lecture 4

Jonathan Dönszelmann
Vivian Roest

Delft University of Technology, The Netherlands

January 24, 2024

Previous Lecture

- Programming without an operating system
- Hardware abstraction
- Unsafe code

Today

- Foreign function interfaces
- Cross compilation
- Sending messages
- Tying up loose ends

The last lecture!

Foreign Function Interfaces

Question:

How can two different programs interoperate?

Foreign Function Interfaces

Question:

How can two different programs interoperate when they are written in a different language?

Foreign Function Interfaces

Question:

How can two different programs interoperate when they are written in a different language?

- Network
- File System
- IPC
- Static or Dynamic Linking

Foreign Function Interfaces

Question:

What kind of challenges do we face?

- Network
- File System
- IPC
- Static or Dynamic Linking

Linking together a program

- Static linking
- Compiler creates object files
- Linker creates a binary from many object files
- Symbol: name of item in an object file
- An object file can be ‘Looking’ for a symbol
- Another object file can provide or declare this symbol

Looking for a symbol in C

```c
// declare that it exists
// don't actually define it
extern void do_thing(int);

int main() {
    do_thing(42);
}
```

Another C file compiled seperately

```c
void do_thing(int a) {
    printf("%d", a);
}
```

The linker will make sure that `do_thing` is resolved in the other object file (if defined!)

C to C++

```
1   // declare that it exists
2   // don't actually define it
3   extern void do_thing(int);
4
5   int main() {
6       do_thing(42);
7   }
```

A C++ file compiled seperately

```
1   void do_thing(int a) {
2       stc::cout << a << std::endl;
3   }
```

C and C++ work similarly, so this is often possible[1].

[1] Is C a subset of C++?

C to C++

Not valid c++

```
1   void* ptr;
2   int* i = ptr;
```

C to C++

Not valid c++

```
1   void example() {
2       goto foo;
3       int i = 1;
4   foo:
5       ;
6   }
```

C to C++

Not valid c++

```
1   void example(int* restrict a, int* restrict b) {
2   }
```

C to Rust

Question:

What kind of problems will we face?

C to Rust

Question:

What kind of problems will we face?

- sizes of integers
- irrepresentable types (enums with values)
- exceptions (panic)
- fat pointers
- incompatible ABI
- generic functions

Impersonating C

```c
// declare that it exists
// don't actually define it
extern void do_thing(int);

int main() {
    do_thing(42);
}
```

Define a symbol with the same name in Rust

```rust
use std::ffi::c_int;

#[no_mangle]
pub extern "C" fn do_thing(a: c_int) {
    println!("{}", a)
}
```

So what's going on here?

```
1    use std::ffi::c_int;
2
3    #[no_mangle]
4    pub extern "C" fn do_thing(a: c_int) {
5        println!("{}", a)
6    }
```

- `#[no_mangle]`: ask Rust not to change the function name[2]
- `extern "C"`: ask Rust to make the function callable from C
- `c_int` is an integer of the same size as an `int` in C.

[2] `https://godbolt.org/z/PaYsv61E5`

ABI

- `extern "C"`: sets an ABI
- default: `extern "rust"`
- ABI: Application Binary Interface
- What does a program expect where to work
- example: what register contains the return value?
- Rust's ABI is pretty unstable
- C's ABI is pretty stable [3]

Remember:

```
1    #[repr(C)]
2    struct A {b: u64}
```

[3] ish, technically it defines no ABI but loads of programs assume it does and it is pretty constant on a single architecture

C to Python

Question:

What kind of extra problems will we face?

C to Python

Question:

What kind of problems will we face?

- Garbage collector
- Interpreted
- Completely different representation of values

Rust to Python

- Just like Rust can simply link to C so can
  - C++
  - Python
  - Java (ish)
  - Pretty much every other language
- So we can use C's ABI to talk to other languages
- PyO3 to talk to Python for example: `docs.rs/pyo3`

Making interaction easier

- `cc` automatically compiles and links C files through cargo
- `build.rs` files can run rust code at compile time to do extra tasks
- `cbindgen` can generate C headers from Rust types

Demo!

Foreign Function Interfaces

Question:

How can two different programs interoperate when they are written in a different language?

- Network
- File System
- IPC
- Static or Dynamic Linking

Sending and Receiving messages

Communication happens over some kind of channel

- Network
- File System
- IPC
- Static or Dynamic Linking

Question:

What kind of data can we send over these channels?

Sending and Receiving messages

- Sending bytes or characters
- We can convert more complex data into bytes that represent them
  - `u32` to 4 bytes
  - *f64* to 8 bytes
  - structs to the concatenated bytes of their contents
  - etc
- This is called serialization, the reverse is called deserialization
- Goal: receiver deserializes to the same information as the sender sent

Question:

How can we serialize references?

Sending and Receiving messages

- Serialization can quickly become hard
- Can we automate this?
- Two steps:
  1. inspecting types (struct/enums/etc)
  2. outputting serialized data
- Serde does part 1: generate code at compile time so they are inspectable at runtime

```rust
use serde::{Serialize, Deserialize};

#[derive(Serialize, Deserialize, PartialEq)]
struct Ping {
    // Some data fields for the ping message
    timestamp: u64,
    payload: Vec<u8>,
}
```

Sending and Receiving messages

Now we can use an external library like `postcard` to convert instances of this struct to bytes

```rust
use postcard::{to_vec, from_bytes};

#[derive(Serialize, Deserialize, PartialEq)]
struct Ping { ... }

let original = Ping {
    timestamp: 0x123456789abcdef,
    payload: vec![0, 1, 2, 3, 4, 5, 6, 7, 8],
};

// ser is just a Vec<u8> representing the original message
let ser = to_vec(&original);

// can fail when our bytes are not a valid Ping message
let de: Ping = from_bytes(ser.deref()).unwrap();

assert_eq!(original, de);
```

Sending and Receiving messages

- Different backends for different output formats
- `postcard` for small binary representations
- `serde_json` to convert types to and from json
- Lots of others: `https://serde.rs/`

Wire protocols

Question

After we serialized, can we just send our messages over a wire (like UART)?

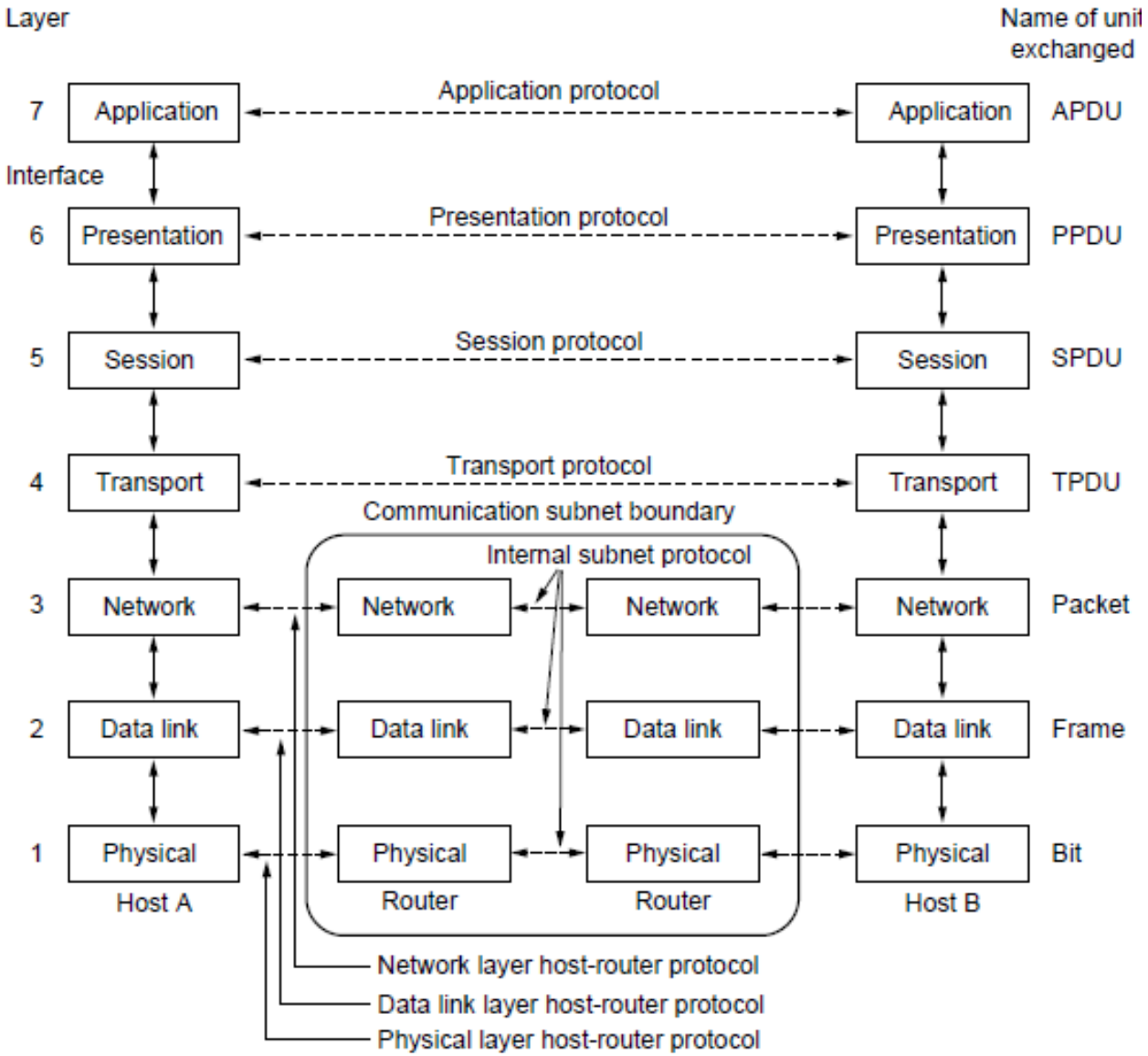
Wire protocols

Question

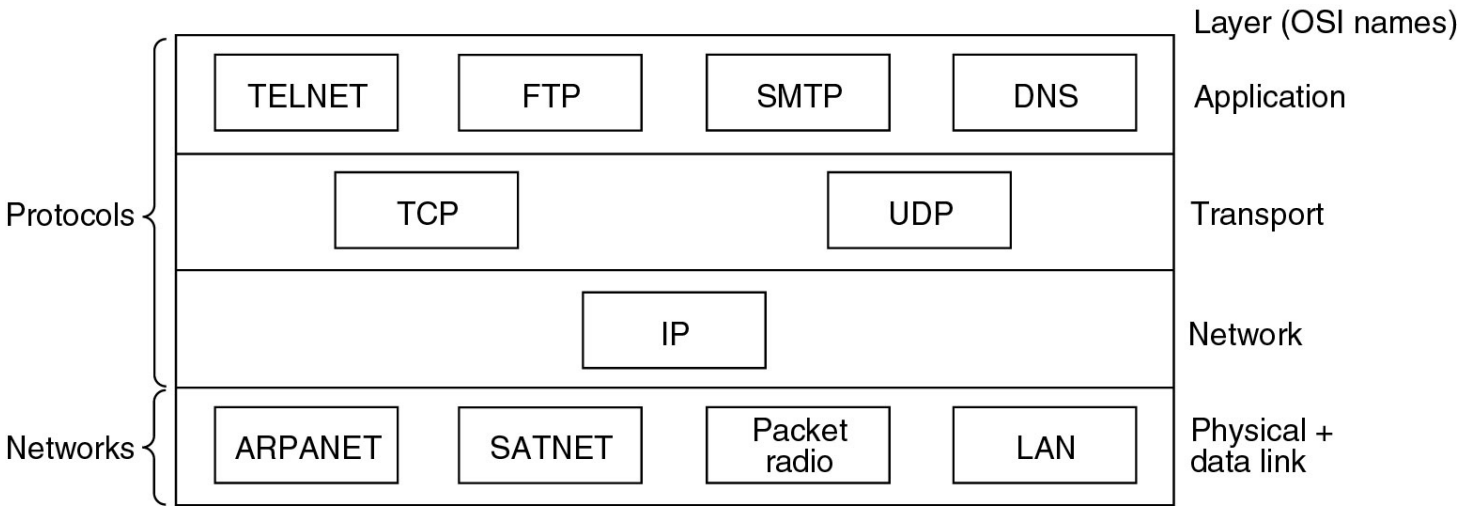
After we serialized, can we just send our messages over a wire (like UART)?

- Where do messages start?
- Start and end markers
- What if that marker occurs in the data we want to send?
- Escaping / byte stuffing
- Prefixing lengths

Networking


Layer
Name of unit exchanged
7
Application
Application protocol
Application
APDU
Interface
6
Presentation
Presentation protocol
Presentation
PPDU
5
Session
Session protocol
Session
SPDU
4
Transport
Transport protocol
Transport
TPDU
Communication subnet boundary
Internal subnet protocol
3
Network
Network
Network
Network
Packet
2
Data link
Data link
Data link
Data link
Frame
1
Physical
Physical
Physical
Physical
Bit
Host A
Router
Router
Host B
Network layer host-router protocol
Data link layer host-router protocol
Physical layer host-router protocol

Networking


Layer (OSI names)
TELNET
FTP
SMTP
DNS
Application
Protocols
TCP
UDP
Transport
IP
Network
Networks
ARPANET
SATNET
Packet radio
LAN
Physical + data link

Data integrity

Question

What happens if a byte is lost or changed while we send it?

Data integrity

Question

What happens if a byte is lost or changed while we send it?

- Checksums
- Like a hash function
- Sent along with the message
- if the receiver finds they don't match, reject the message or ask for retransmission
- commonly used: CRC

Cross compilation

- Compiling on one system, *for* another system
- The other system could have a different OS, different architecture etc.
- The compiling system must know the details of the target system to know what code to generate
- Target triples: `<arch>-<vendor>-<system>-<ABI>`
- example:
  - `x86_64-pc-windows-gnu`
  - `x86_64-pc-windows-msvc`
  - `x86_64-unknown-linux-gnu`
  - `riscv32i-unknown-none-elf`

Question:

Why do we need cross compilation?

Cross compilation

Question:

Why do we need cross compilation?

- Some target systems don't have an operating system, how can we run compilers on them?
- Sometimes it's just easier to compile on a different system: better hardware?

Demo! `https://doc.rust-lang.org/nightly/rustc/platform-support.html`

End of part 1

This is the end of our part of this course, we hope you enjoyed it!