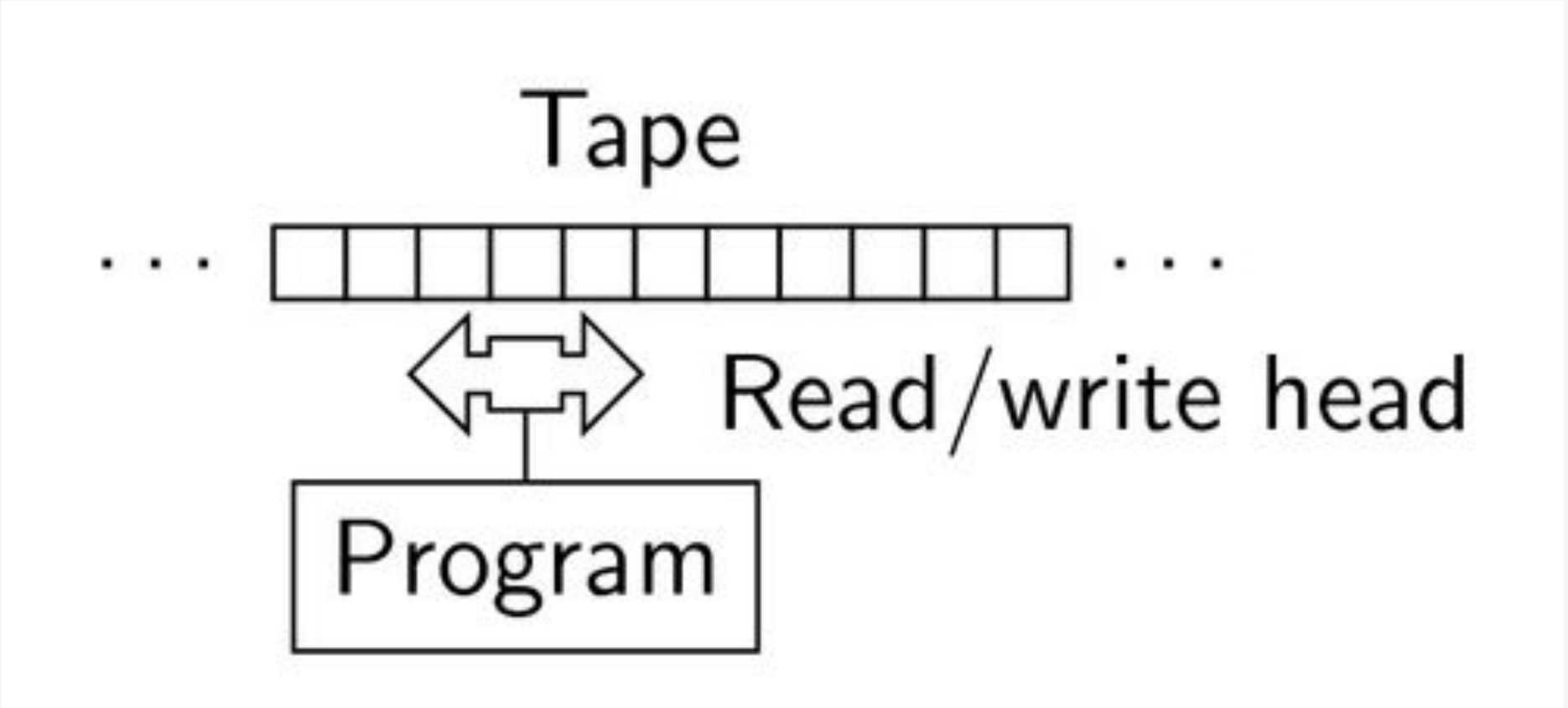
Turing Complete EBPF

John Fastabend, Isovalent

Turing Machine: Cartoon



Turing Complete: Colloquial Test

- Loops: While(), goto, ...
- Control flow (if, else)
- Unlimited Memory

Turing Complete: Colloquial Test

- Loops: While(), goto, ...
- Control flow (if, else)
- ~~Unlimited~~ **Arbitrary Large** Memory

Turing Complete: What does it actually mean

- Cannot Solve:
  - halting problem
  - mortality problem
  - word problem for groups

- Can:
  - simulate Turing machines (emulation, simulation)
  - program: Game of Life Rule110
  - compute general-recursive functions

Turing Complete: What does it actually mean

- Cannot Solve:
  - halting problem
  - mortality problem
  - word problem for groups

Do I need to solve any of these problems in BPF?

- Can:
  - Parse L4/L7 protocols
  - DDOS Protections
  - Collect system metrics
  - File Integrity Monitoring

Who Cares? 3 characters

- Character 1: Learning BPF and/or trying to understand big picture
  - How do I map BPF onto languages/runtimes I know?
  - What are BPF common use cases?
  - Should I invest in BPF?
- Character 2:
  - BPF is not as powerful as X because of Turing completeness
  - BPF can not compute this Turing Complete things Foo
  - BPF can not solve my use case because of verification
- Character 3:
  - Fun and games

For many use cases
bounded runtime is a good property

Upper bound running time is useful for many use case

- Networking
- Kernel security
- File Integrity Monitoring
- Scheduler?
- XDP
- ….

Challenge: Create an event based system where unbounded runtime is a useful property?
Did you want to run that use case in the Linux kernel?

But for fun lets
show how to make BPF Turing Complete

- Arbitrary large memory: maps
- Control Flow: normal C control flow
- Loops: ???

DoAgain

```
do_again(val, func) {
        bpf_timer_set_callback(&val→timer, func);
        bpf_timer_start(&val→timer, 0, 0);
}

static int func(void *map, int *key, struct v *val)
```

How to show BPF is Turing Complete

- To show turing complete implement Turing Machine
- Turing Machines are a bit of a pain to code
- It is good enough to implement something that is equivalent to ™
- 2-tag systems are TM

2-Tag System

Production Rules

A = ABAB
B = A
…

Cut Number

C = 2

Input String

ABAABBBBB
AABBBBBABAB
BBBBBBABAB
BBBBABABA
BBABABAA
ABABAAA
ABAAAABAB
….

DEMO

Production Rules

A = ABAB
B = A
…

Cut Number

C = 2

Input String

ABAABBBBB
AABBBBBABAB
BBBBBBABAB
BBBBABABA
BBABABAA
ABABAAA
ABAAAABAB
….

Questions/Comments

Should have a blog post soon.