Flexible Anonymous Network

Florentin Rochet, Olivier Bonaventure, and Olivier Pereira

UCLouvain Crypto Group, Belgium

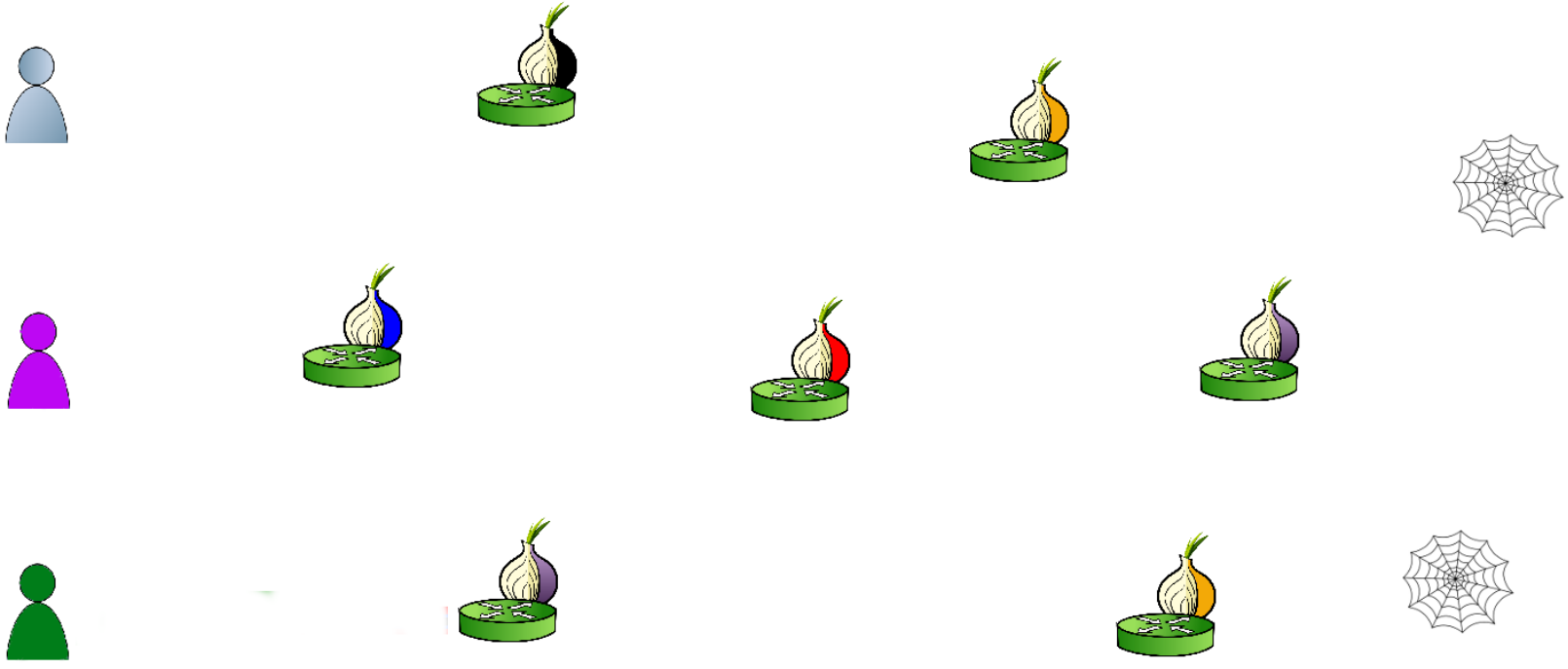
UCLouvain IP Networking Lab, Belgium

Tor

- A distributed network run by volunteers to seperate identification from the routing task

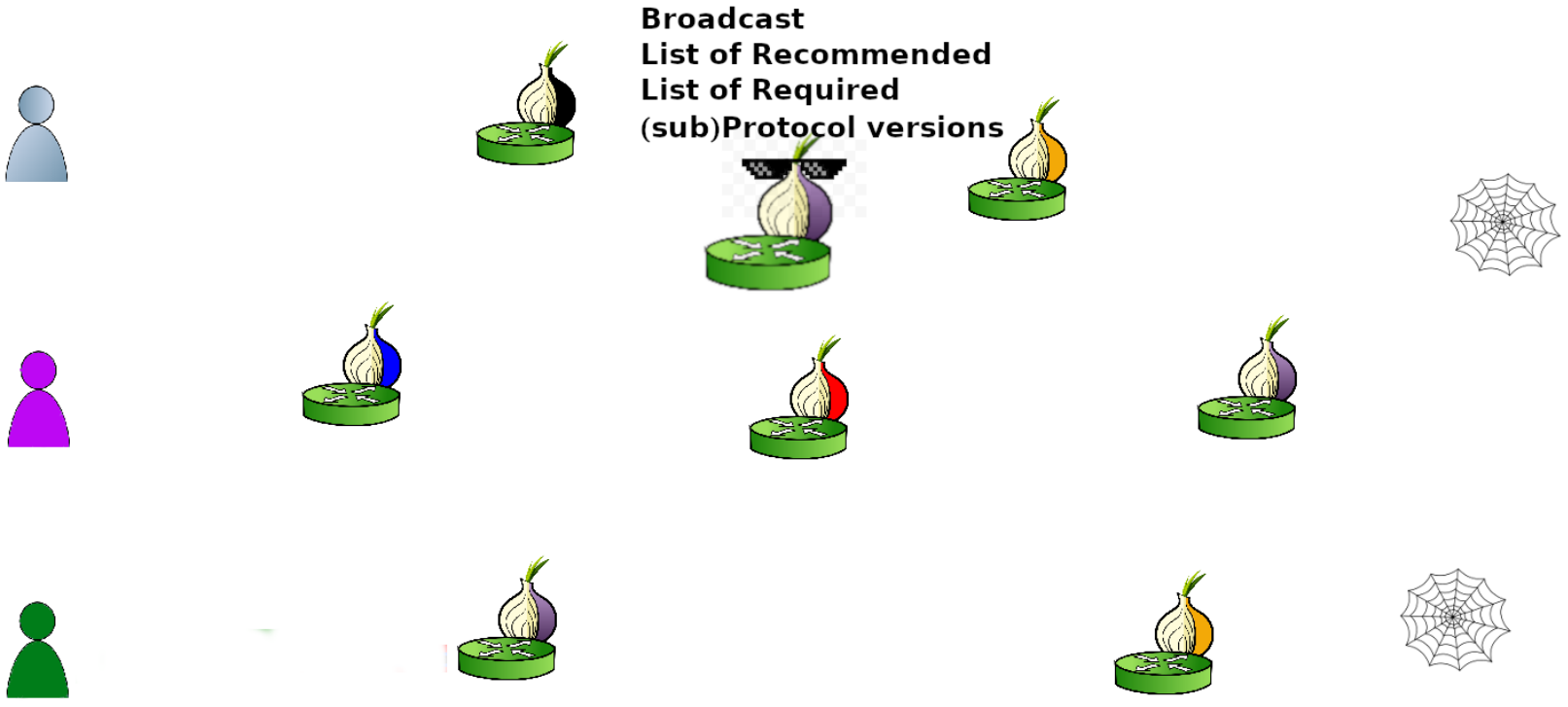
Features deployment

- Deploying new protocol features is painfully difficult



Features deployment

- Deploying new protocol features is painfully difficult



The impossible choice

It is also about security!

- Protocol tolerance (as implemented today) is a vector to efficient attacks[1, 2, 3, 4, 5]

1: "Dropping on the Edge: Flexibility and Traffic Confirmation in Onion Routing Protocols", PoPETs 2018

2: "CMU-FBI relay_early confirmation attack", (see Tor's blog post)

3: "The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network", NDSS 2014

4: "Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization", S&P 2013

5: "A Practical Congestion Attack on Tor Using Long Paths", Usenix Security 2009

We need to deploy fixes faster

... without excluding any relay from the network

(Probably impossible with *current* deployment methods)

Introducing FAN

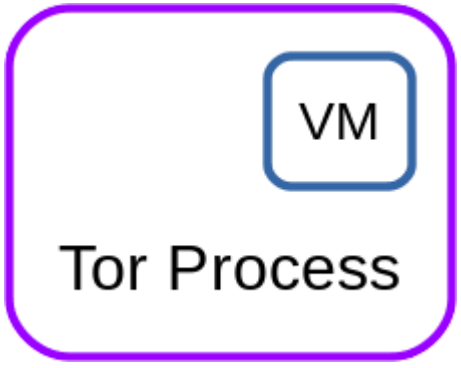
Definition:

> *We call FAN, for Flexible Anonymous Network, an anonymous network architecture able to transparently change its behavior for one or many users without having to restart relays or perturbing other user connections while proceeding to add, remove or modify protocol features.*

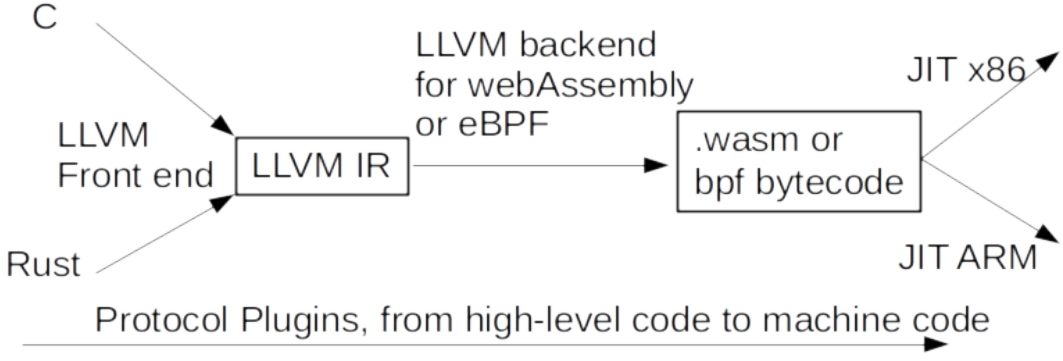
- Threat model is context-dependent (we will see why)
- High performance

What is the magic trick?

A userland VM



- Run within the Tor process
- Implements a RISC architecture
- Load and execute **"Protocol Plugins"** (upon bytecode authentication)
- Protocol Plugins are sandboxed



Hello World!

code in hello_world.c:

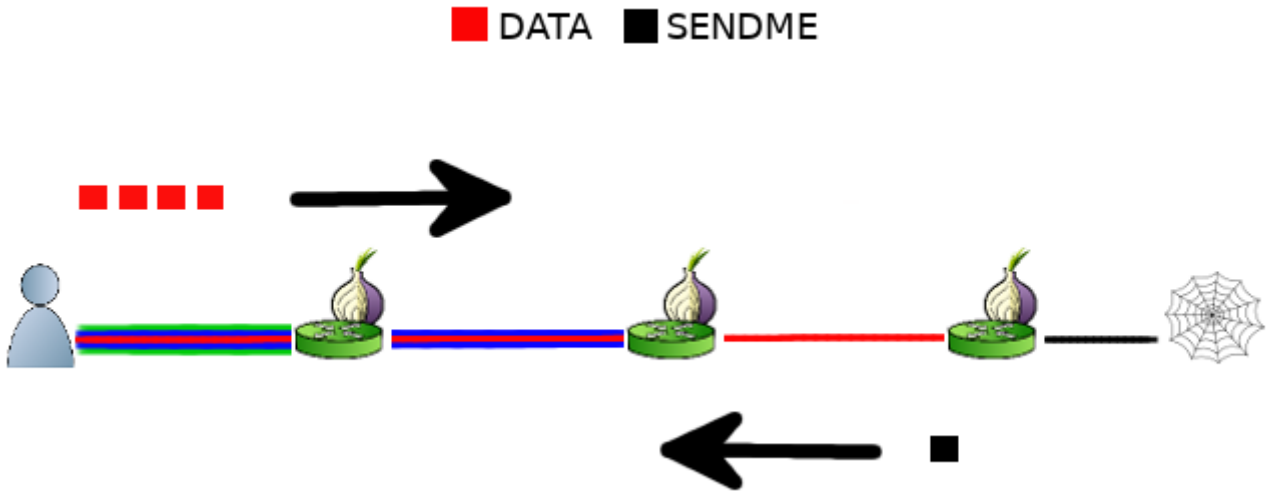
```
#include "core/or/plugin.h"
// things that can be defined in a .h  and included here
#include "hello_world_features.h"
// My plugin main entry point
uint64_t hello_world(void *args) {
  log_fn_(LOG_DEBUG, LD_PLUGIN, __FUNCTION__,
    "Hello, I am becoming self-aware. Run.");
  return 0;
}
```

Meta-info in hello_world.plugin:

```
hello_world replace some_tor_function for_some_module hello_w
```

How would Protocol Plugins impact performance on a real usecase

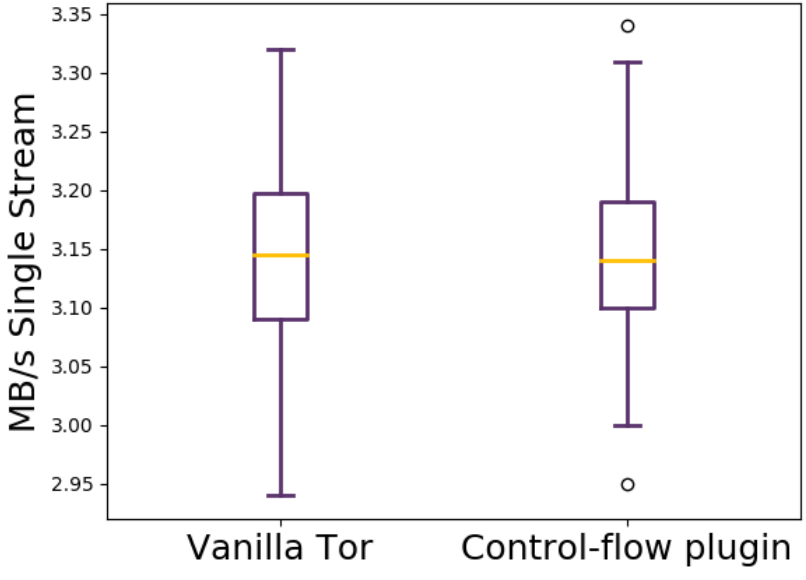
10000ft flow-control overview

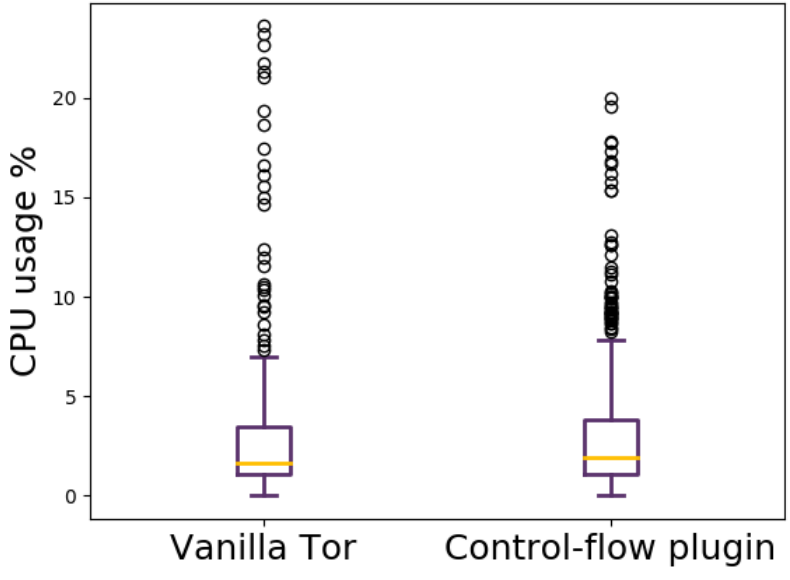


- Is versionned (new version currently in deployment)
- New version solves fairness and security issues, but would take many years to be widely used
  - Deployment could be almost instantaneous with Protocol Plugins

SENDME cells in a plugin

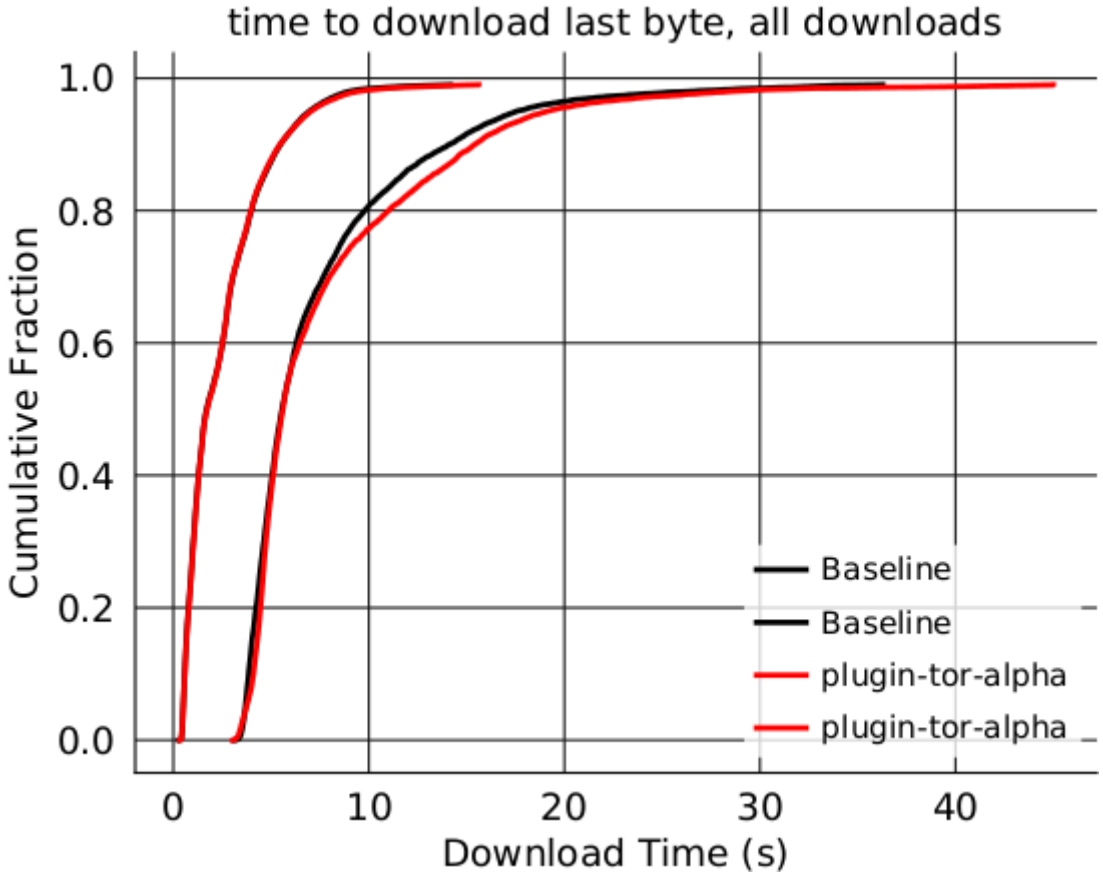
4 nodes (client-relay-relay-relay) on the loopback (4 cpus); 20 MB stream pushed 50 times





Some perf eval

200 relays, 2000 clients:



How to properly integrate?

- What *should* be extensible? (ongoing research)
- What about safety and security for a network-wide extension system? (ongoing research)
  - Safety: sending protocol plugins to the whole network *must* be a multi-dev agreement
  - Security: threshold signatures (TUF?[1]); *must* survive key compromise;
- Is eBPF the right tool? What about webAssembly? (ongoing research)
- Advancing Tor's control over plugin execution (ongoing research)

[1] J. Samuel, N. Mathewson, J. Cappos, and R. Dingledine. Survivable key compromise in software update systems. In Proceedings of the 17th ACM conference on Computer and communications security, pages 61–72. ACM, 2010

Custom Internet Privacy (Further Work)

- Can we go further than re-designing forward compatibility?
- What if clients plug their own set of features to their ephemeral connection?
  - ✓ Could improve performance/anonymity tradeoff (ongoing research)
    - e.g., Plug a padding scheme when using a given .onion
    - e.g., Join a mixnet plugin when sending emails
  - ✓ We could push the threat model to the application (or even to the user for expert mode)
  - ✓ Protocol Plugins could ease contributions from the research community
  - ✗ Huh. Great remote code exploitation toolset, what can go wrong?
    - 1 piece of the puzzle to defend in our upcomming ACM SIGCOMM'19 "Pluginizing QUIC" work

Conclusion

- Protocol Plugins is a generic solution, and may be used to address many problems
  - e.g., censorship? Using an authorized application supporting protocol plugins to hide ephemeral features (e.g., end-to-end secure messaging over bitcoin gossiping protocol?)
  - ... many more ;)
- Custom Internet Privacy: the quest for the one anonymous network that fits many usages!
- 10+ years of research ahead with theoretical and practical challenges!
  - Getting security right is going to take time
- Disclaimer: current VM implementation is experimental and has some strong limitations
  - But heh, that would eventually be much improved

Be conservative in what you do, stay conservative in what you accept from others