TCP/IP fingerprint *emulation.*

Using user-space networking and Rust

JOE DYE (HE/HIM) · BYTEFUL

Agenda

01	What is TCP/IP *fingerprinting?*
02	How is TCP/IP fingerprinting *done?*
03	How *might* TCP/IP fingerprinting done?
04	How do we *emulate* TCP/IP fingerprints?

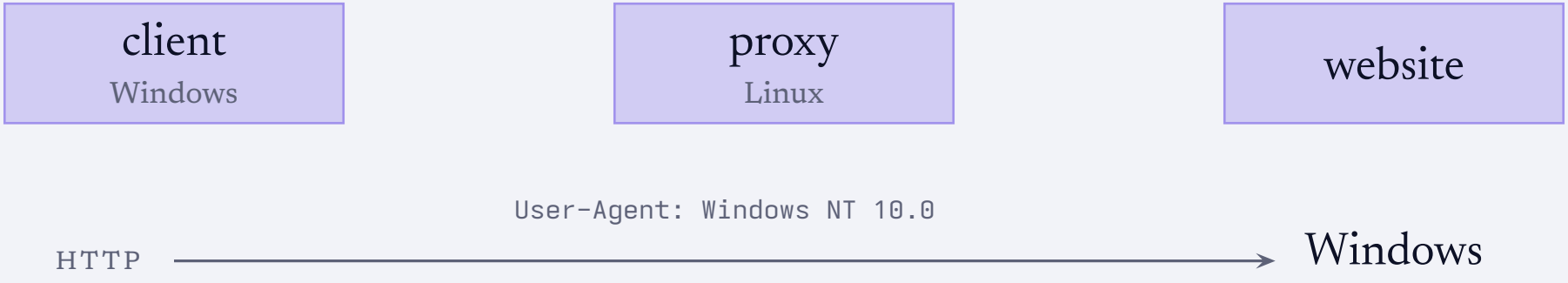
01

What is TCP/IP *fingerprinting?*

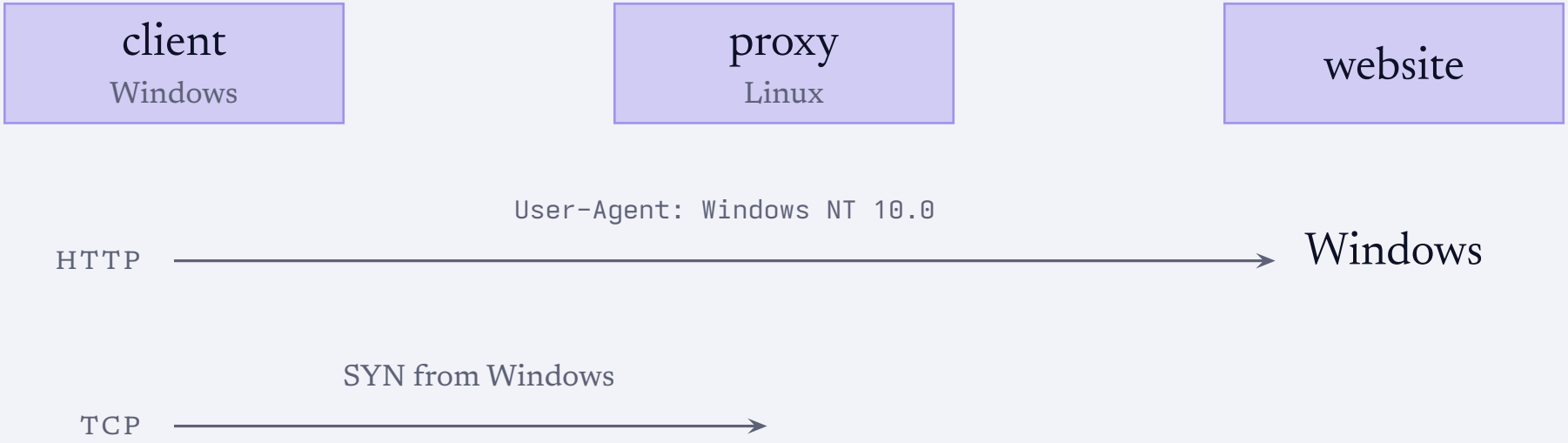
Introduction



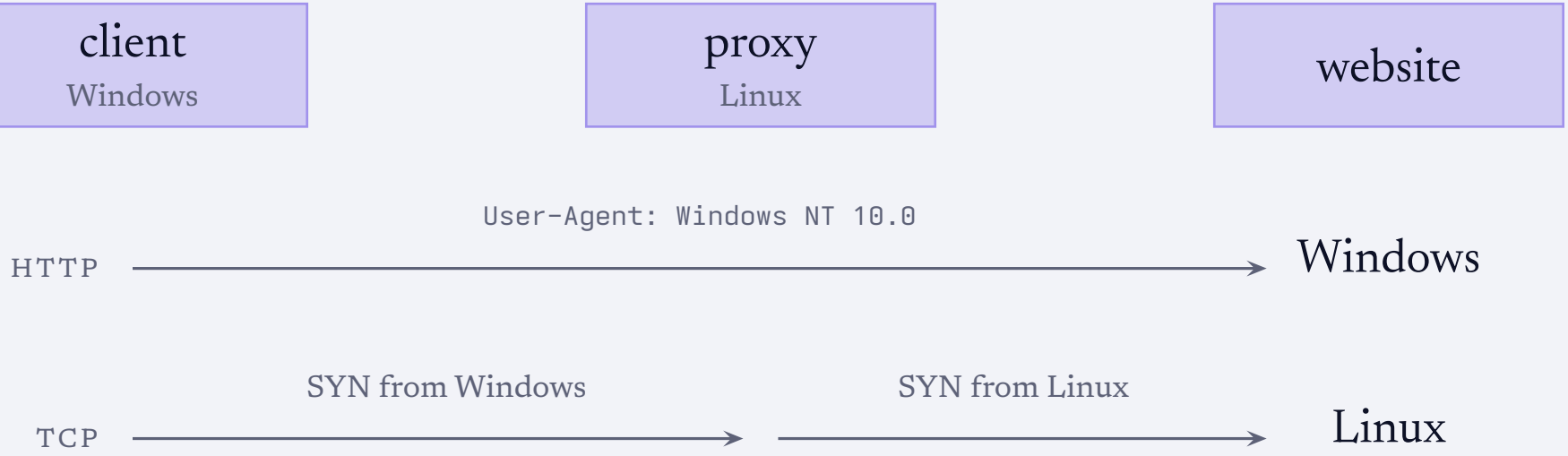
Introduction



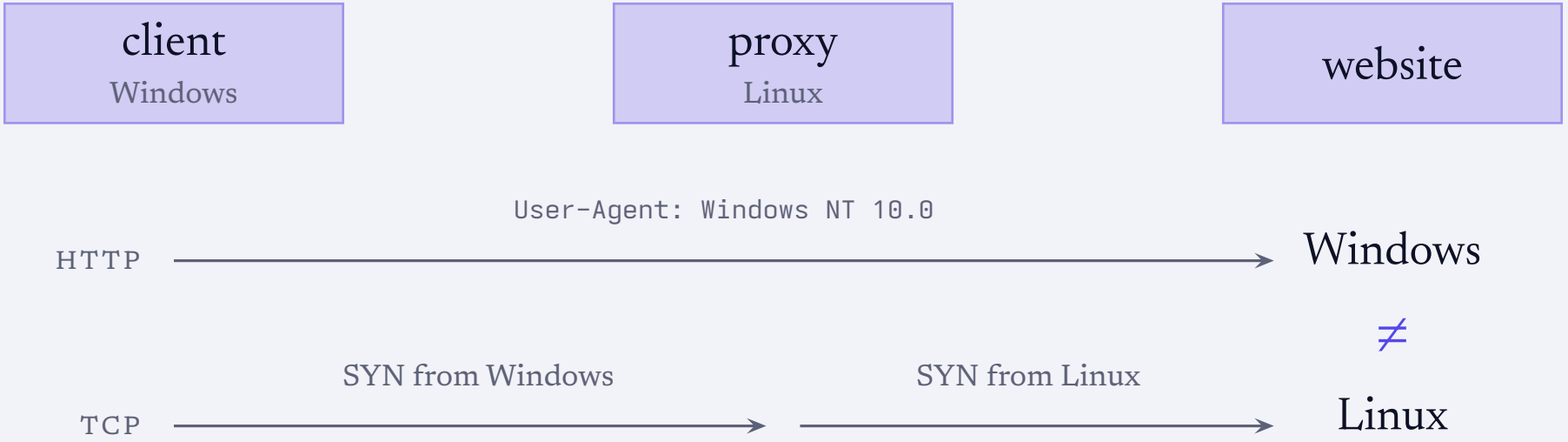
Introduction



Introduction



Introduction



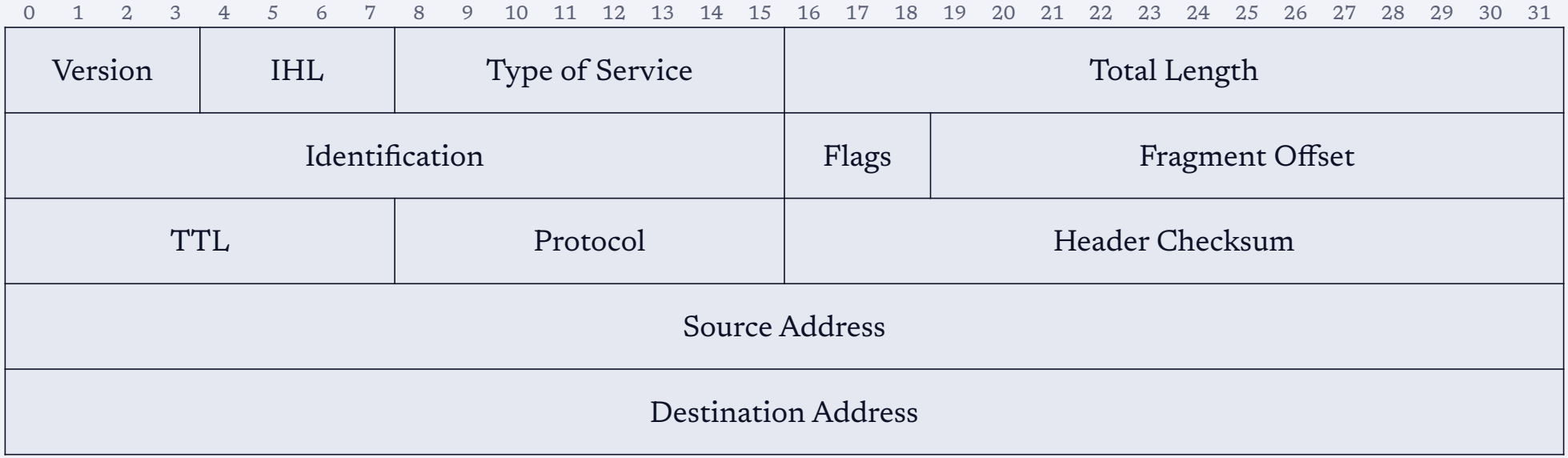
A *mismatch* between the declared-OS and the detected-OS is a strong sign that a proxy is being used.

Origins

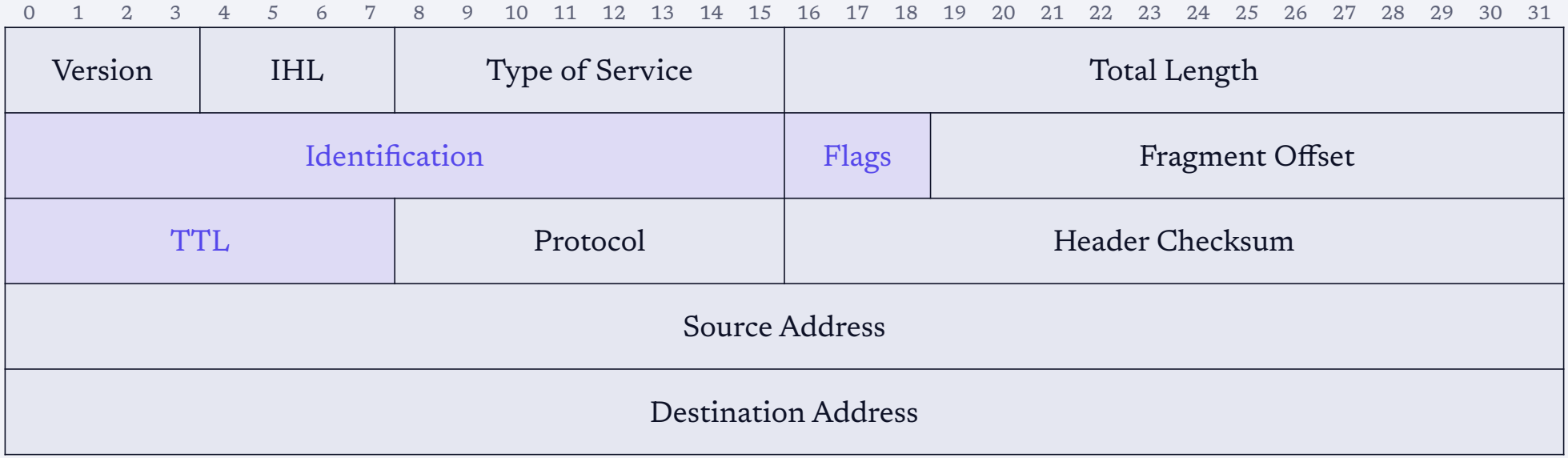


Looks for *quirks* in the implementation an operating system provides.

IP Header


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
TTL
Protocol
Header Checksum
Source Address
Destination Address

IP Header



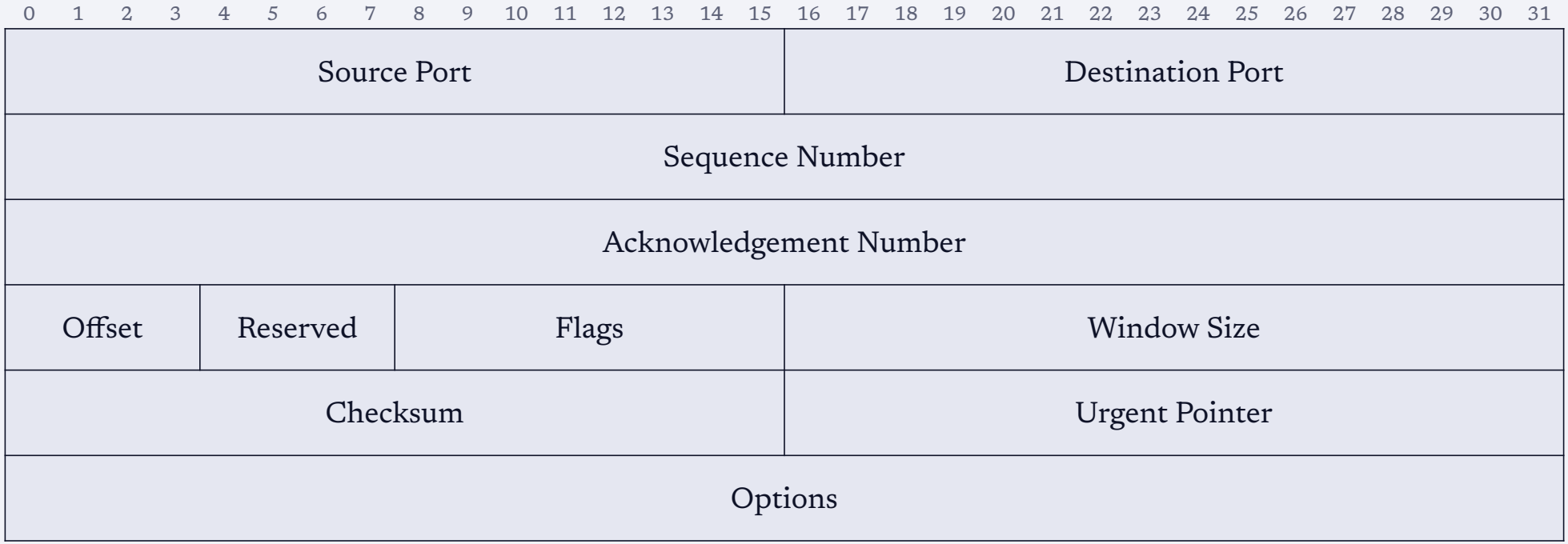
■ OS-correlated

IP Headers

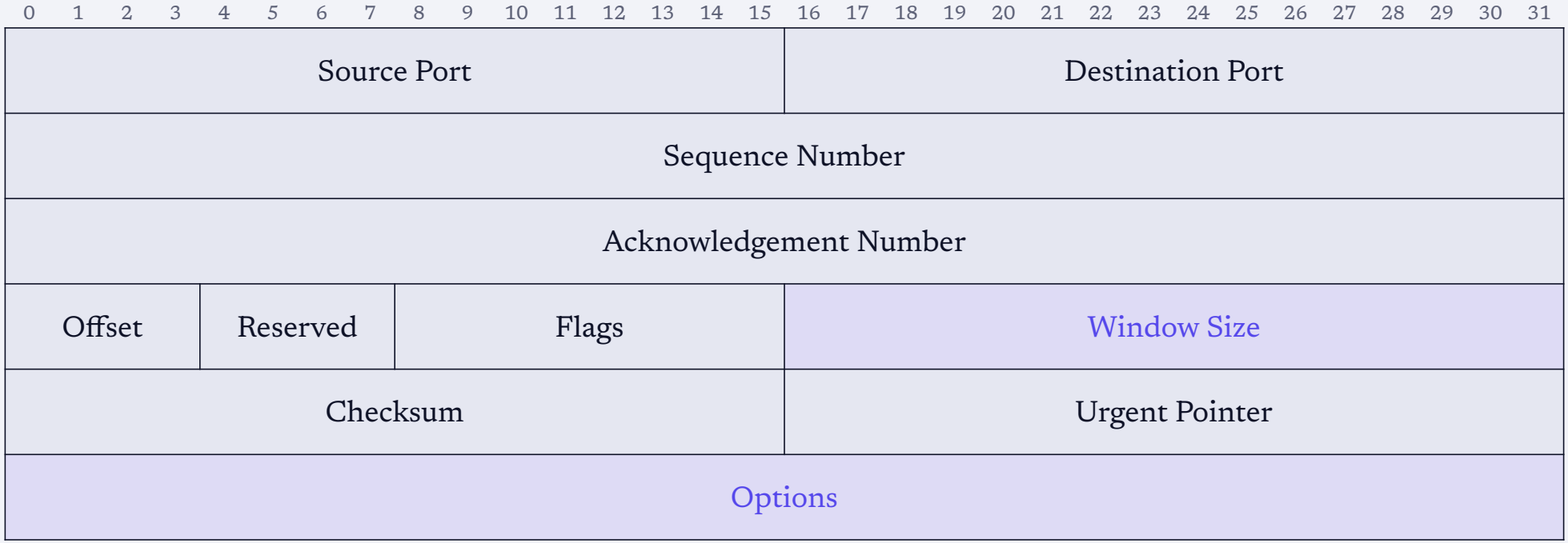
IHL	Header length
Length	Total packet length
ID	Fragmentation identifier
Flags	Don't Fragment (DF) and More Fragments (MF) fields
TTL	Number of allowed hops
Protocol	Encapsulated protocol - always 6 for TCP
Checksum	Error checking

■ OS-correlated

TCP Header


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Source Port
Destination Port
Sequence Number
Acknowledgement Number
Offset
Reserved
Flags
Window Size
Checksum
Urgent Pointer
Options

TCP Header



■ OS-correlated

TCP Headers

SEQ	Randomised starting number for packet sequence
ACK	Always 0 in SYN
Data Offset	Length of TCP headers
Window Size	Maximum buffer for receiving data
Checksum	Error checking
Urgent PTR	Always 0 in SYN
Options	Extensions to the protocol for improved performance

■ OS-correlated

Default Values by OS

OS	TTL	Window	Scale	DF	IP ID
Linux	64	64240	7	set	per flow
Android	64	65535	8	set	per flow
Windows	128	64240	8	set	per target
macOS	64	65535	6	set	none
iOS	64	65535	6	set	none

Default Values by OS

OS	TTL	Window	Scale	DF	IP ID
Linux	64	64240	7	set	per flow
Android	64	65535	8	set	per flow
Windows	128	64240	8	set	per target
macOS	64	65535	6	set	none
iOS	64	65535	6	set	none

Default Values by OS

OS	TTL	Window	Scale	DF	IP ID
Linux	64	64240	7	set	per flow
Android	64	65535	8	set	per flow
Windows	128	64240	8	set	per target
macOS	64	65535	6	set	none
iOS	64	65535	6	set	none

Default Values by OS

OS	TTL	Window	Scale	DF	IP ID
Linux	64	64240	7	set	per flow
Android	64	65535	8	set	per flow
Windows	128	64240	8	set	per target
macOS	64	65535	6	set	none
iOS	64	65535	6	set	none

Default Values by OS

OS	TTL	Window	Scale	DF	IP ID
Linux	64	64240	7	set	per flow
Android	64	65535	8	set	per flow
Windows	128	64240	8	set	per target
macOS	64	65535	6	set	none
iOS	64	65535	6	set	none

Try It Yourself...

M1452-S-T-N-W7	M1412-N-W6-N-N-T-S-E-E
M1452-N-W6-N-N-T-S-E-E	M1460-S-T-N-W7
M1460-S-T-N-W12	M1452-N-W6-N-N-T-S-E-E
M1400-S-T-N-W8	M1460-N-W6-N-N-T-S-E-E
M1460-N-W6-N-N-T-S-E-E	M1452-S-T-N-W9
M1410-S-T-N-W7	M1360-N-W6-N-N-T-S-E-E

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Try It Yourself...

?
M1452-S-T-N-W7
M1452-N-W6-N-N-T-S-E-E
M1460-S-T-N-W12

?
M1412-N-W6-N-N-T-S-E-E
M1460-S-T-N-W7
M1452-N-W6-N-N-T-S-E-E

?
M1400-S-T-N-W8
M1460-N-W6-N-N-T-S-E-E
M1410-S-T-N-W7

?
M1460-N-W6-N-N-T-S-E-E
M1452-S-T-N-W9
M1360-N-W6-N-N-T-S-E-E

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Try It Yourself...

Linux
M1452-S-T-N-W7
M1460-S-T-N-W7
M1410-S-T-N-W7

Mac OS
M1452-N-W6-N-N-T-S-E-E
M1452-N-W6-N-N-T-S-E-E
M1460-N-W6-N-N-T-S-E-E

Android
M1460-S-T-N-W12
M1400-S-T-N-W8
M1452-S-T-N-W9

iOS
M1412-N-W6-N-N-T-S-E-E
M1460-N-W6-N-N-T-S-E-E
M1360-N-W6-N-N-T-S-E-E

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

Apple	M1412 – N-W6 – N-N-T – S-E-E
Linux	M1410 – S-T – N-W7
Windows	M1460 – N-W8 – N-N-S

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

Apple	M1412 – N-W6 – N-N-T – S-E-E
Linux	M1410 – S-T – N-W7
Windows	M1460 – N-W8 – N-N-S

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

Apple	M1412 – N-W6 – N-N-T – S-E-E
Linux	M1410 – S-T – N-W7
Windows	M1460 – N-W8 – N-N-S

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

Apple	`M1412 – N-W6 – N-N-T – S-E-E`
Linux	`M1410 – S-T – N-W7`
Windows	`M1460 – N-W8 – N-N-S`

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

Apple	`M1412 – N-W6 – N-N-T – S-E-E`
Linux	`M1410 – S-T – N-W7`
Windows	`M1460 – N-W8 – N-N-S`

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

Apple	M1412 – N-W6 – N-N-T – S-E-E
Linux	M1410 – S-T – N-W7
Windows	M1460 – N-W8 – N-N-S

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

Apple	`M1412 – N-W6 – N-N-T – S-E-E`
Linux	`M1410 – S-T – N-W7`
Windows	`M1460 – N-W8 – N-N-S`

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

Apple	`M1412 – N-W6 – N-N-T – S-E-E`
Linux	`M1410 – S-T – N-W7`
Windows	`M1460 – N-W8 – N-N-S`

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

Apple	`M1412 – N-W6 – N-N-T – S-E-E`
Linux	`M1410 – S-T – N-W7`
Windows	`M1460 – N-W8 – N-N-S`

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

Apple	`M1412 – N-W6 – N-N-T – S-E-E`
Linux	`M1410 – S-T – N-W7`
Windows	`M1460 – N-W8 – N-N-S`

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

Option Ordering

Apple	`M1412 – N-W6 – N-N-T – S-E-E`
Linux	`M1410 – S-T – N-W7`
Windows	`M1460 – N-W8 – N-N-S`

M: Maximum Segment Size **W**: Window Scaling **N**: No-op **S**: Selective ACK **T**: TCP Timestamp **E**: End of Options

02

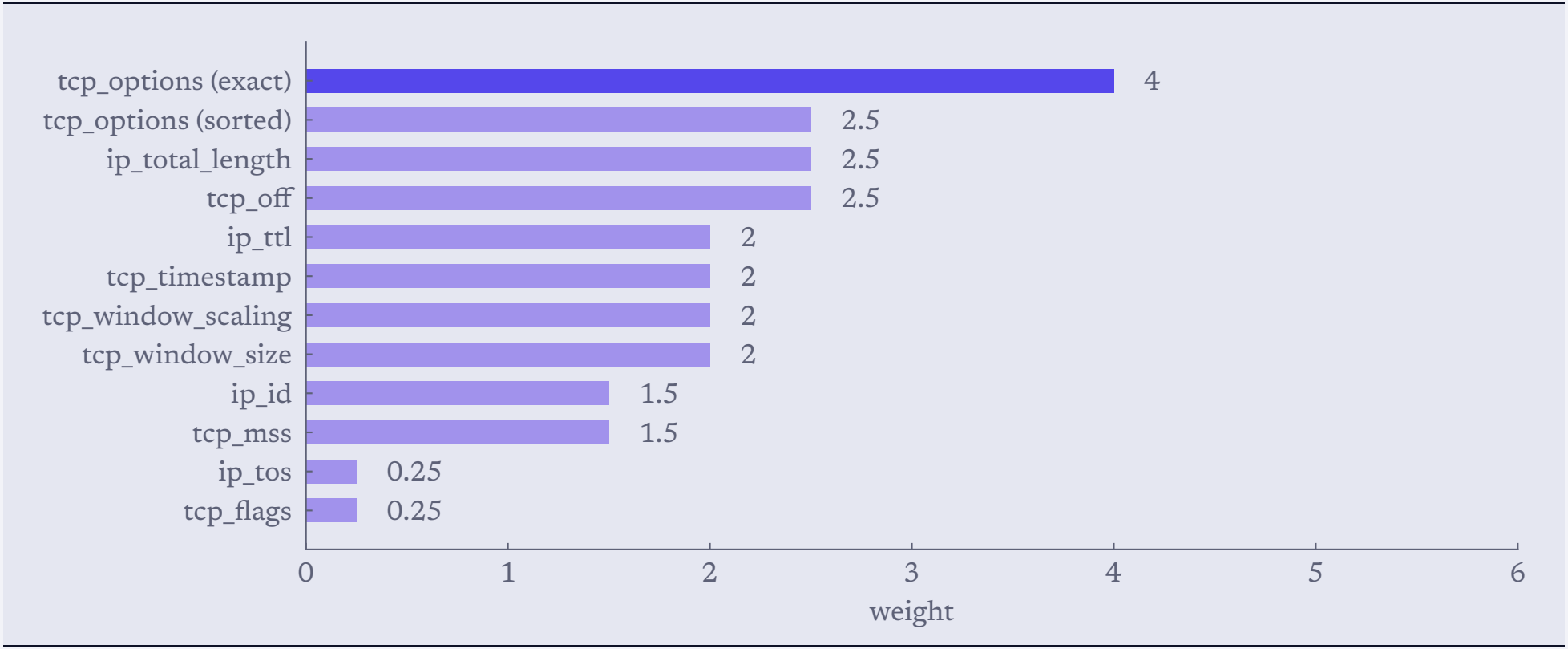
How is TCP/IP fingerprinting *done?*

Zardaxt

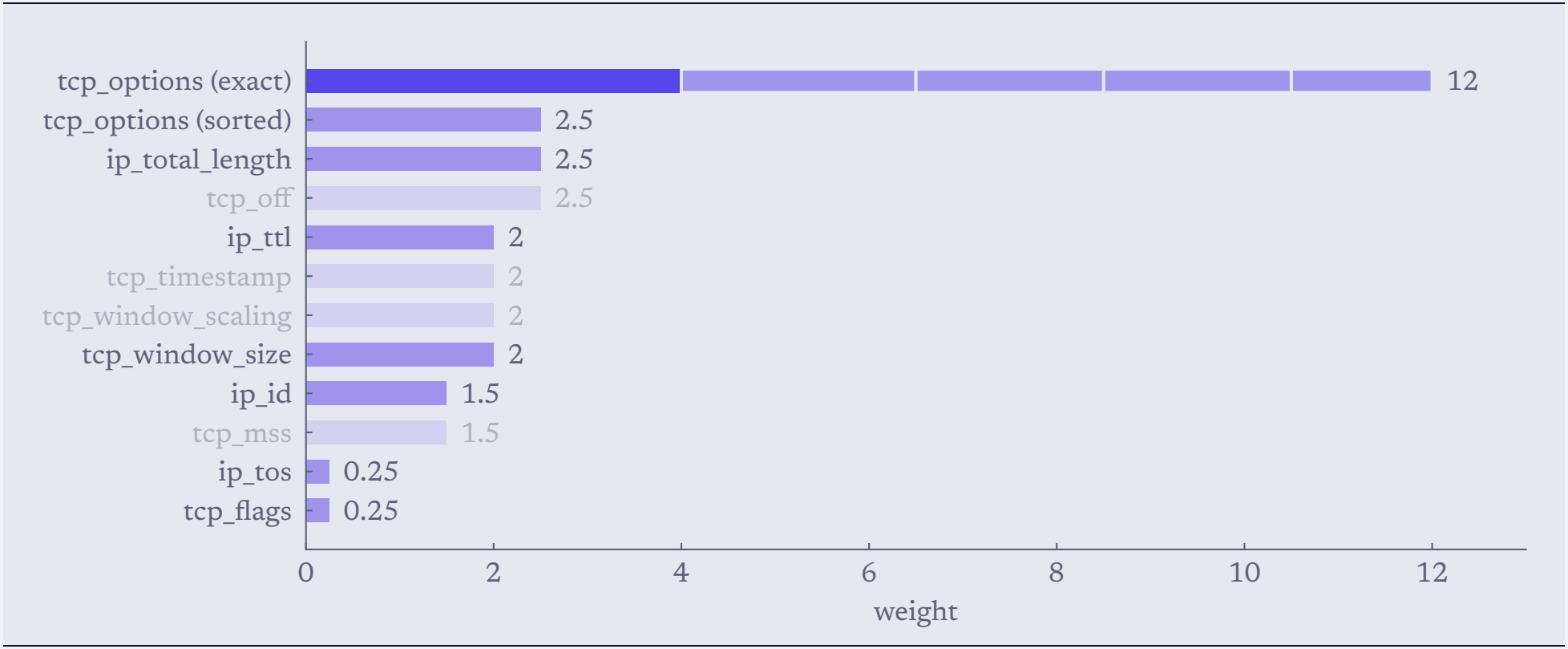
```
score = 0
os_name = entry['os']

if entry['ip_id'] == fp['ip_id']:
    score += 1.5
if entry['ip_tos'] == fp['ip_tos']:
    score += 0.25
if entry['ip_total_length'] == fp['ip_total_length']:
    score += 2.5
if entry['ip_ttl'] == fp['ip_ttl']:
    score += 2
if entry['tcp_off'] == fp['tcp_off']:
    score += 2.5
if entry['tcp_timestamp_echo_reply'] == fp['tcp_timestamp_echo_reply']:
    score += 2
if entry['tcp_window_scaling'] == fp['tcp_window_scaling']:
    score += 2
if entry['tcp_window_size'] == fp['tcp_window_size']:
    score += 2
if entry['tcp_flags'] == fp['tcp_flags']:
    score += 0.25
if entry['tcp_mss'] == fp['tcp_mss']:
    score += 1.5
if entry['tcp_options'] == fp['tcp_options']:
    score += 4
elif entry['tcp_options_ordered'] == fp['tcp_options_ordered']:
    score += 2.5
```

Zardaxt


tcp_options (exact)
tcp_options (sorted)
ip_total_length
tcp_off
ip_ttl
tcp_timestamp
tcp_window_scaling
tcp_window_size
ip_id
tcp_mss
ip_tos
tcp_flags
4
2.5
2.5
2.5
2
2
2
2
1.5
1.5
0.25
0.25
0
1
2
3
4
5
6
weight

Zardaxt


tcp_options (exact)
tcp_options (sorted)
ip_total_length
tcp_off
ip_ttl
tcp_timestamp
tcp_window_scaling
tcp_window_size
ip_id
tcp_mss
ip_tos
tcp_flags
12
2.5
2.5
2.5
2
2
2
2
1.5
1.5
0.25
0.25
0
2
4
6
8
10
12
weight

Huginn-Net

```rust
#[derive(Clone, Debug, PartialEq)]
pub enum Quirk {
    Df,                     // df     - "don't fragment" set (probably PMTUD)
    NonZeroID,              // id+    - DF set but IPID non-zero
    ZeroID,                 // id-    - DF not set but IPID is zero
    Ecn,                    // ecn    - explicit congestion notification support
    MustBeZero,             // 0+     - "must be zero" field not zero
    FlowID,                 // flow   - non-zero IPv6 flow ID
    SeqNumZero,             // seq-   - sequence number is zero
    AckNumNonZero,          // ack+   - ACK number non-zero, ACK flag not set
    AckNumZero,             // ack-   - ACK number zero, ACK flag set
    NonZeroURG,             // uptr+  - URG pointer non-zero, URG flag not set
    Urg,                    // urgf+  - URG flag used
    Push,                   // pushf+ - PUSH flag used
    OwnTimestampZero,       // ts1-   - own timestamp specified as zero
    PeerTimestampNonZero,   // ts2+   - non-zero peer timestamp on initial SYN
    TrailingNonZero,        // opt+   - trailing non-zero data in options
    ExcessiveWindowScaling, // exws   - excessive window scaling (> 14)
    OptBad,                 // bad    - malformed TCP options
}
```

Huginn-Net Signature

```
4 : 64+0 : 0 : 1460 : mss*44,7 : mss,sok,ts,nop,ws : df,id+ : 0
```

Huginn-Net Signature

```
4 : 64+0 : 0 : 1460 : mss*44,7 : mss,sok,ts,nop,ws : df,id+ : 0
```

IP version

Huginn-Net Signature

`4` : `64+0` : `0` : `1460` : `mss*44,7` : `mss,sok,ts,nop,ws` : `df,id+` : `0`

TTL + hops

Huginn-Net Signature

```
4 : 64+0 : 0 : 1460 : mss*44,7 : mss,sok,ts,nop,ws : df,id+ : 0
```

IP options length

Huginn-Net Signature

`4` : `64+0` : `0` : `1460` : `mss*44,7` : `mss,sok,ts,nop,ws` : `df,id+` : `0`

MSS

Huginn-Net Signature

`4` : `64+0` : `0` : `1460` : `mss*44,7` : `mss,sok,ts,nop,ws` : `df,id+` : `0`

window size, scale

Huginn-Net Signature

`4` : `64+0` : `0` : `1460` : `mss*44,7` : `mss,sok,ts,nop,ws` : `df,id+` : `0`

option layout

Huginn-Net Signature

`4` : `64+0` : `0` : `1460` : `mss*44,7` : `mss,sok,ts,nop,ws` : `df,id+` : `0`

quirks

Huginn-Net Signature

`4` : `64+0` : `0` : `1460` : `mss*44,7` : `mss,sok,ts,nop,ws` : `df,id+` : `0`

payload class

03

How *might* TCP/IP fingerprinting be done?

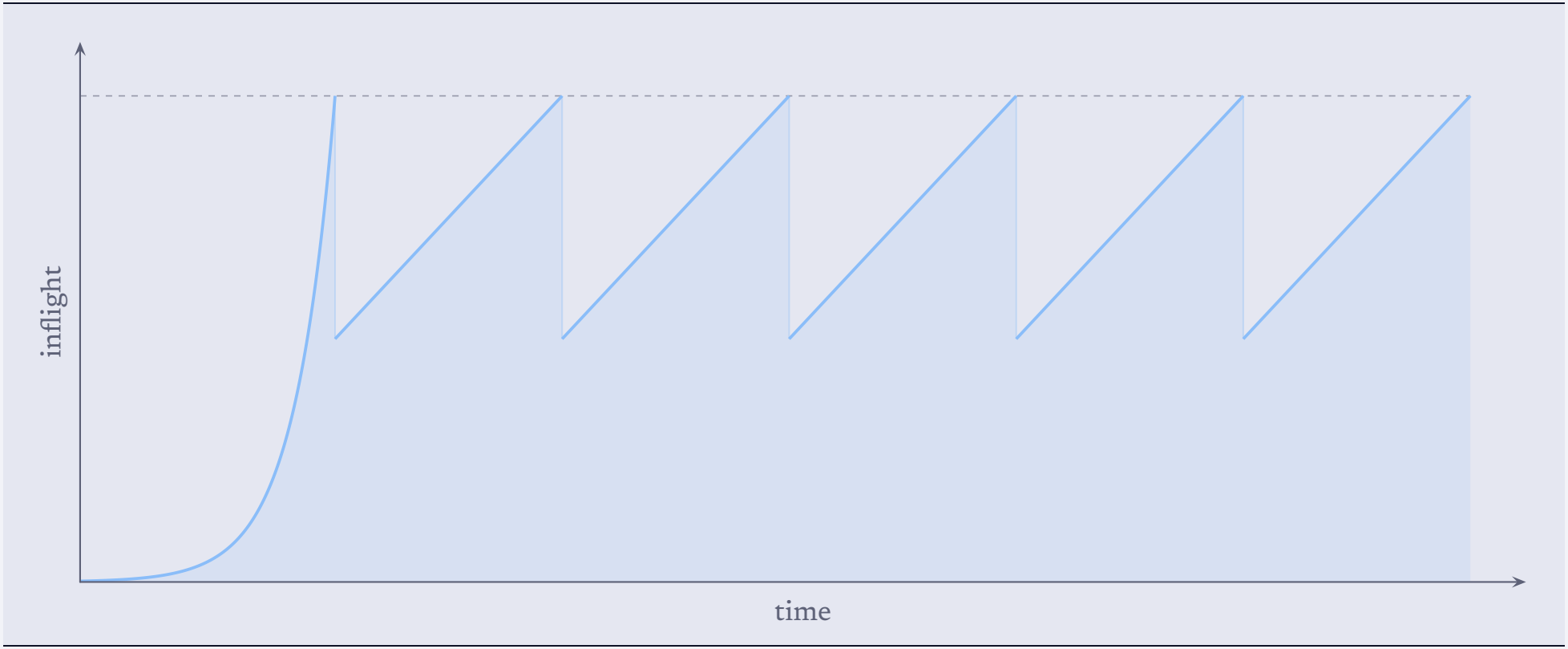
Future Tools

- Congestion control algorithm.
- Buffer growth algorithm.
- Fragmentation ID generation algorithm.
- Timestamp generation algorithm.
- Minimum retransmission timeout.

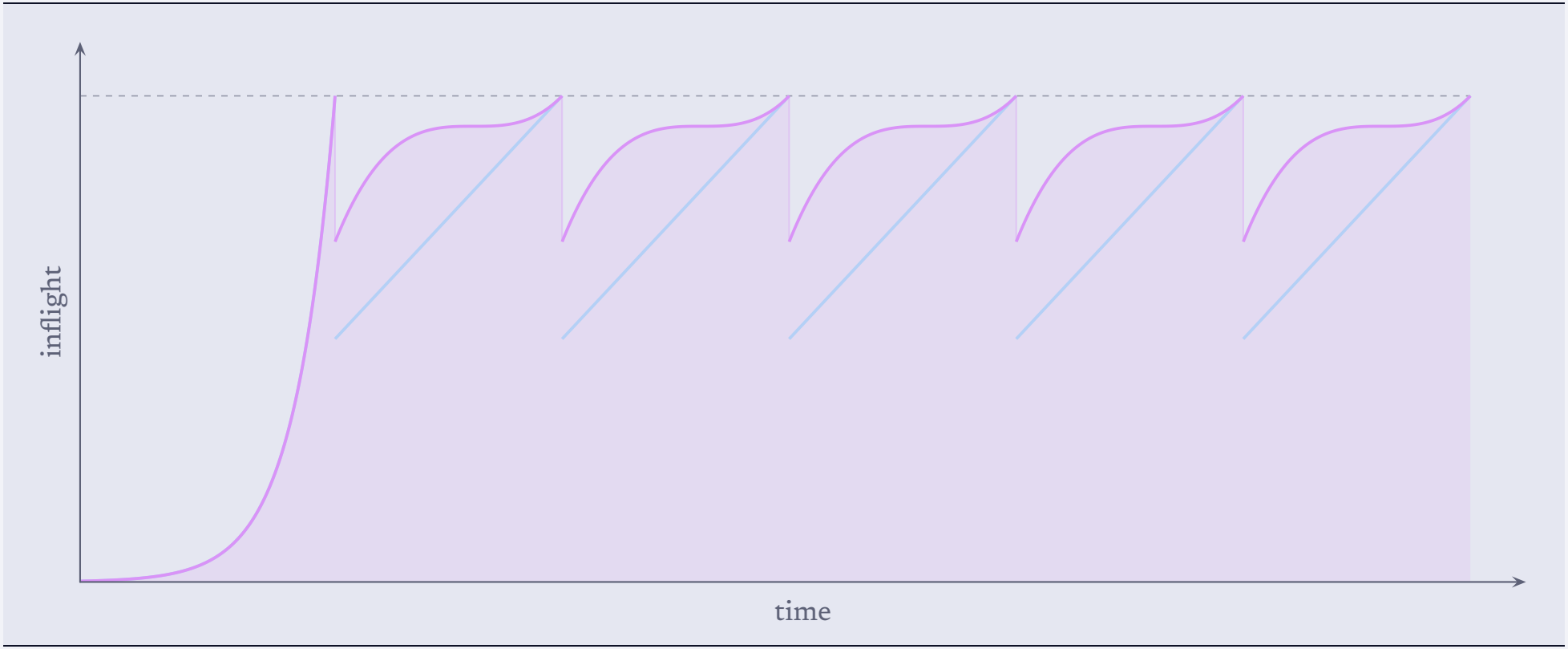
Congestion Control



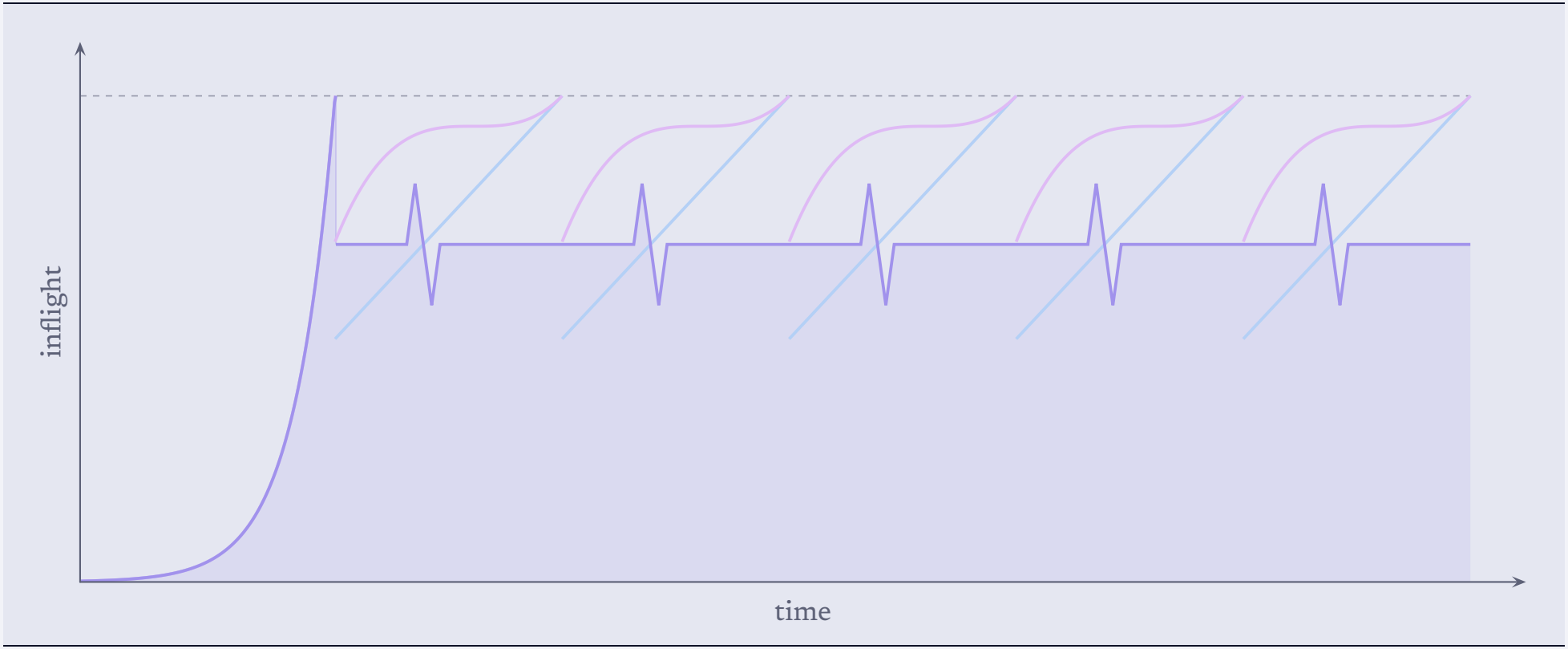
Congestion Control



Congestion Control



Congestion Control



IP ID Generation

OS	Generator
Linux, Android	per flow
Windows	per target
macOS, iOS	none

04

How do we *emulate* TCP/IP fingerprints?

Requirements

fast	Parity with default Linux TCP stack.
portable	Run on servers and on personal devices.
maintainable	I have other things to do with my time.
flexible	Don’t know what we don’t know.
dynamic	Specify OS in proxy credentials.

Possible Approaches

Custom Kernel	eBPF	User-Space TCP
+ Quickest MVP	+ Is cool	+ Total control
+ Have experience	+ Super fast	+ Very portable
– Not portable	– Limited scope	– TCP is hard
– I hate C	– Still C	– TCP is very hard

Possible Approaches

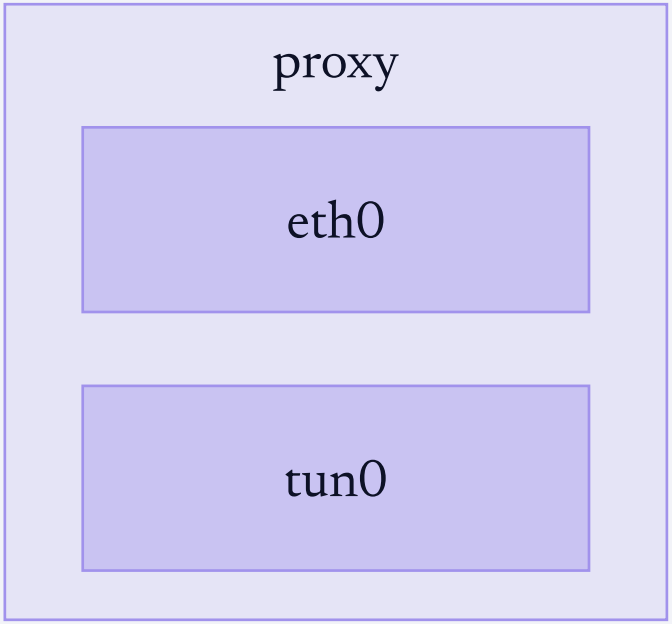
Custom Kernel	eBPF	User-Space TCP
+ Quickest MVP	+ Is cool	+ Total control
+ Have experience	+ Super fast	+ Very portable
– Not portable	– Limited scope	– TCP is hard
– I hate C	– Still C	– TCP is very hard

Only solution that meets all criteria is a custom TCP/IP stack

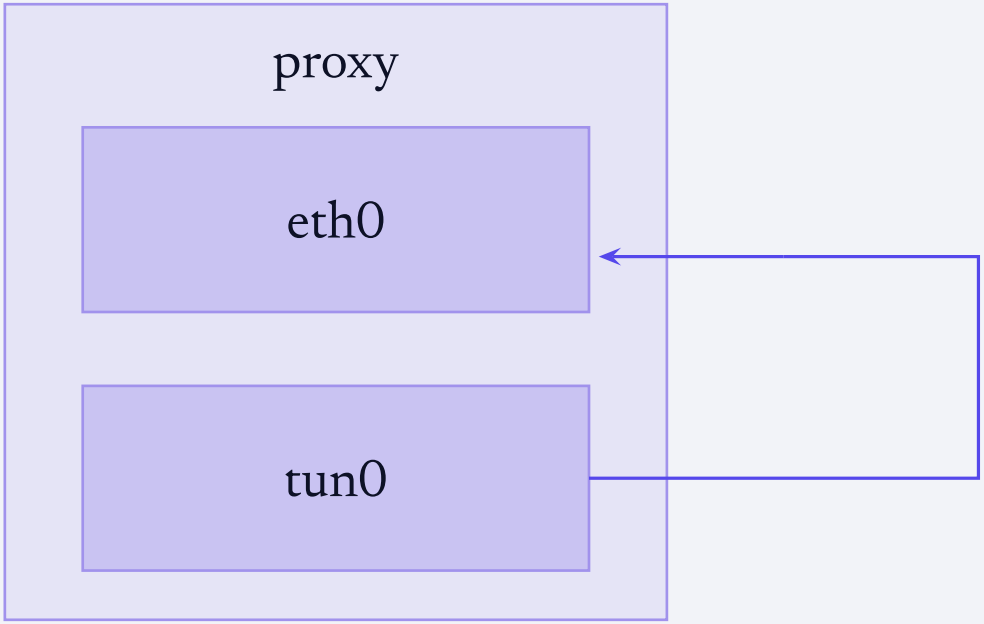
User-Space Networking

Bypass	How it works
Raw sockets	Send and receive bytes by bypassing some of the kernel's TCP/IP stack.
TUN/TAP	A virtual NIC that bypasses most of the kernel's TCP/IP stack.
DPDK	Write our own poll-mode drivers and get/put bytes directly off/on the NIC.

TUN/TAP

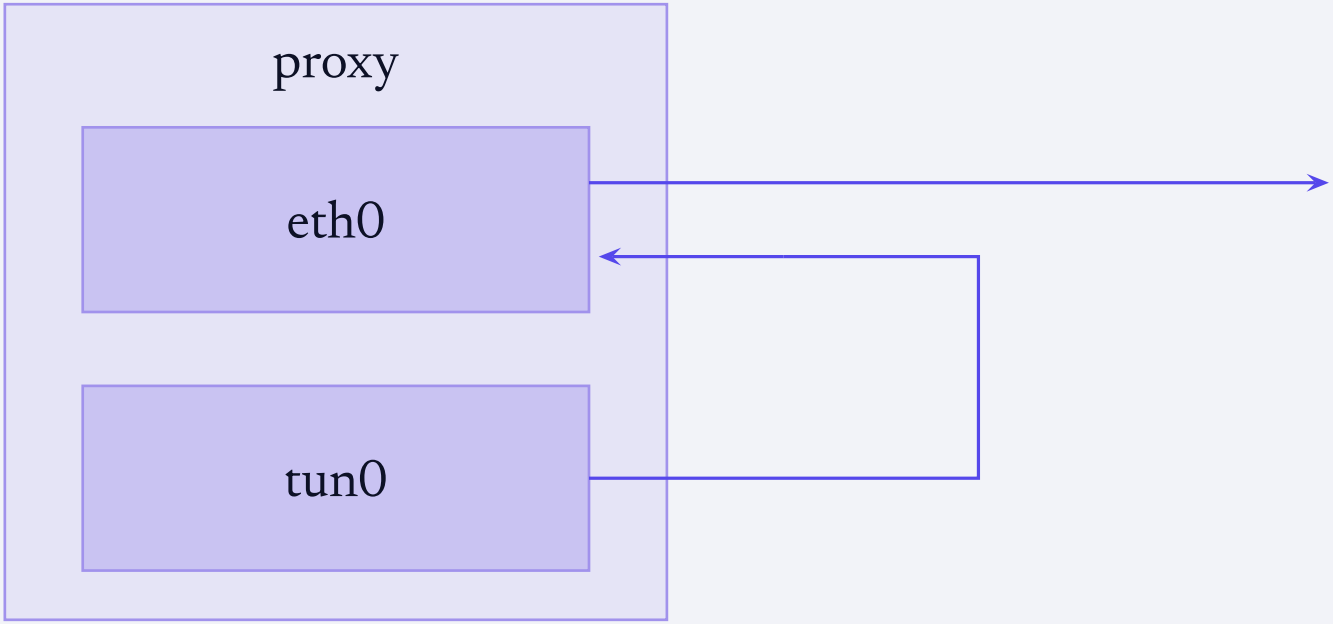


TUN/TAP



The OS thinks the packet came from *another machine*.

TUN/TAP



The OS thinks the packet came from *another machine*.

smoltcp

> smoltcp is a standalone, event-driven TCP/IP stack that is designed for bare-metal, real-time systems. Its design goals are simplicity and robustness.

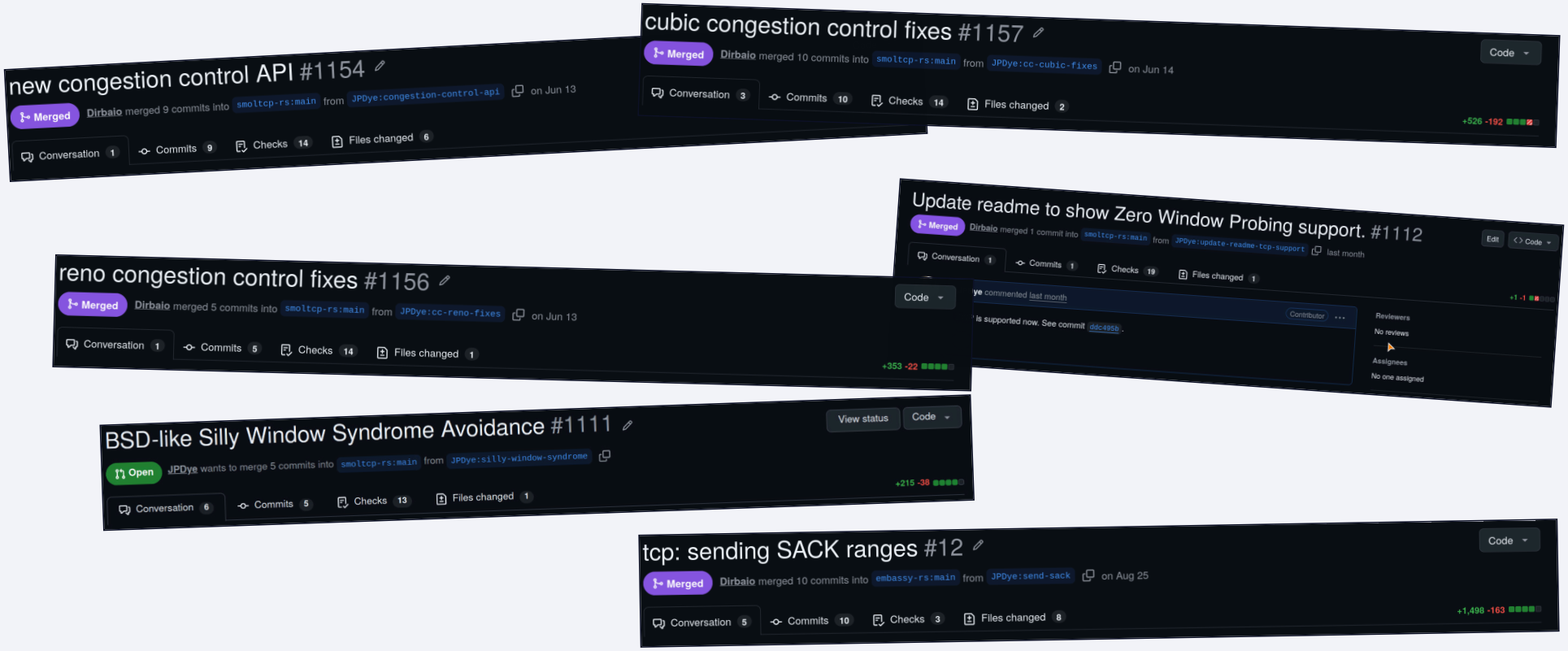
```
    .-[ 127.0.0.1/4321 -> 127.0.0.1/1234 (syn) ]-
|
| client   = 127.0.0.1/4321
| os       = ???
| dist     = 0
| params   = none
| raw_sig  = 4:64+0:0:65495:mtu*1,0:mss,ws,sok,eol+2,eol+1,eol+0:df:0
```

smoltcp

smoltcp is *missing* many widely deployed features.

- Selective acknowledgements.
- Timestamping.
- Congestion control.
- Buffer growth.
- Zero-window probes.
- Silly Window Syndrome avoidance.

PRs


new congestion control API #1154
Merged
Dirbaio merged 9 commits into smoltcp-rs:main from JPDye:congestion-control-api on Jun 13
Conversation 1
Commits 9
Checks 14
Files changed 6
cubic congestion control fixes #1157
Merged
Dirbaio merged 10 commits into smoltcp-rs:main from JPDye:cc-cubic-fixes on Jun 14
Code
Conversation 3
Commits 10
Checks 14
Files changed 2
+526 -192
Update readme to show Zero Window Probing support. #1112
Merged
Dirbaio merged 1 commit into smoltcp-rs:main from JPDye:update-readme-tcp-support last month
Edit
Code
Conversation 1
Commits 1
Checks 19
Files changed 1
+1 -1
commented last month
Contributor
is supported now. See commit ddc495b.
Reviewers
No reviews
Assignees
No one assigned
reno congestion control fixes #1156
Merged
Dirbaio merged 5 commits into smoltcp-rs:main from JPDye:cc-reno-fixes on Jun 13
Code
Conversation 1
Commits 5
Checks 14
Files changed 1
+353 -22
BSD-like Silly Window Syndrome Avoidance #1111
View status
Code
Open
JPDye wants to merge 5 commits into smoltcp-rs:main from JPDye:silly-window-syndrome
Conversation 6
Commits 5
Checks 13
Files changed 1
+215 -38
tcp: sending SACK ranges #12
Code
Merged
Dirbaio merged 10 commits into embassy-rs:main from JPDye:send-sack on Aug 25
Conversation 5
Commits 10
Checks 3
Files changed 8
+1,498 -163

Performance

flows	avg thru	min thru	max thru	fairness	drops
1	133%	133%	133%	0%	-100%
2	3526%	3620%	3439%	0%	-100%
4	1543%	5199%	1538%	14%	-90%
16	1039%	1212%	311%	84%	-98%
32	945%	2386%	543%	28%	-98%
64	753%	1910%	437%	20%	-98%

TCP Option Ordering

```rust
pub struct TcpOptionGroup {
    options: [Option<TcpOptionKind>; MAX_GROUP_OPTIONS],
    pad: TcpOptionPad,
}
```

```rust
pub enum TcpOptionKind {
    Mss,
    WindowScale,
    SackPermitted,
    SackRange,
    Timestamp,
}
```

```rust
pub enum TcpOptionPad {
    LeadNop,
    TrailEol,
}
```

TCP Option Ordering

M1410 - S-T - N-W7

```
const LINUX_SYN: TcpSynOptionLayout = TcpOptionLayoutBuilder::new()
```

TCP Option Ordering

M1410 – S–T – N–W7

```
const LINUX_SYN: TcpSynOptionLayout = TcpOptionLayoutBuilder::new()
    .group(&[TcpOptionKind::Mss], TcpOptionPad::LeadNop)
```

TCP Option Ordering

`M1410 - S-T - N-W7`

```
const LINUX_SYN: TcpSynOptionLayout = TcpOptionLayoutBuilder::new()
    .group(&[TcpOptionKind::Mss], TcpOptionPad::LeadNop)
    .group(
        &[TcpOptionKind::SackPermitted, TcpOptionKind::Timestamp],
        TcpOptionPad::LeadNop,
    )
```

TCP Option Ordering

`M1410 - S-T - N-W7`

```
const LINUX_SYN: TcpSynOptionLayout = TcpOptionLayoutBuilder::new()
    .group(&[TcpOptionKind::Mss], TcpOptionPad::LeadNop)
    .group(
        &[TcpOptionKind::SackPermitted, TcpOptionKind::Timestamp],
        TcpOptionPad::LeadNop,
    )
    .group(&[TcpOptionKind::WindowScale], TcpOptionPad::LeadNop)
    .syn();
```

TCP Option Ordering

```
const LINUX_SYN: TcpSynOptionLayout = TcpOptionLayoutBuilder::new()
    .group(&[TcpOptionKind::Mss], TcpOptionPad::LeadNop)
    .group(&[TcpOptionKind::SackPermitted, TcpOptionKind::Timestamp], TcpOptionPad::LeadNop)
    .group(&[TcpOptionKind::WindowScale], TcpOptionPad::LeadNop)
    .syn();
```

```
const APPLE_SYN: TcpSynOptionLayout = TcpOptionLayoutBuilder::new()
    .group(&[TcpOptionKind::Mss], TcpOptionPad::LeadNop)
    .group(&[TcpOptionKind::WindowScale], TcpOptionPad::LeadNop)
    .group(&[TcpOptionKind::Timestamp], TcpOptionPad::LeadNop)
    .group(&[TcpOptionKind::SackPermitted], TcpOptionPad::TrailEol)
    .syn();
```

```
const WINDOWS_SYN: TcpSynOptionLayout = TcpOptionLayoutBuilder::new()
    .group(&[TcpOptionKind::Mss], TcpOptionPad::LeadNop)
    .group(&[TcpOptionKind::WindowScale], TcpOptionPad::LeadNop)
    .group(&[TcpOptionKind::SackPermitted, TcpOptionKind::Timestamp], TcpOptionPad::LeadNop)
    .syn();
```

TCP Option Ordering

```
fn walk_options(
  groups: &[Option<TcpOptionGroup>],
  p: &TcpOptionPresence,
  values: Option<&Repr<'_>>,
  mut out: Option<&mut [u8]>,
) -> usize {
    // ...
}
```

TCP Congestion Control Algorithm

```rust
pub(super) trait Controller {
    fn window(&self) -> usize;

    fn set_remote_window(&mut self, remote_window: usize) {}
    fn set_mss(&mut self, mss: usize) {}
    fn set_initial_window(&mut self, iw: usize) {}

    fn on_ack(&mut self, now: Instant, len: usize, in_flight: usize, rtt: &RttEstimator) { }
    fn on_dup_ack(&mut self, now: Instant, len: usize, in_flight: usize) {}

    fn on_rto(&mut self, now: Instant, in_flight: usize) {}
    fn on_loss(&mut self, now: Instant, in_flight: usize) {}

    fn pre_transmit(&mut self, now: Instant) {}
    fn post_transmit(&mut self, now: Instant, len: usize) {}

}
```

TCP Congestion Control Algorithm

```rust
pub struct Reno {
  cwnd: usize,
  mss: usize,
  ssthresh: usize,
  rwnd: usize,

  in_fast_recovery: bool,
  in_rto_recovery: bool,
}
```

```rust
pub struct Cubic {
  cwnd: usize,
  mss: usize,
  ssthresh: usize,
  rwnd: usize,
  k: f64,
  w_est: f64,
  cwnd_prior: usize,

  recovery_start: Option<Instant>,
  in_fast_recovery: bool,
  in_rto_recovery: bool,
  app_limited_since: Option<Instant>,
}
```

IP ID Generation

```rust
pub trait IpIdGeneratorFactory {
    fn create(&self, conn: &Connection) -> Box<dyn IpIdGenerator + Send>;
}

pub trait IpIdGenerator {
    fn generate(&self, fragmented: bool) -> u16;
}
```

IP ID Generation

```
// Linux, Android: a new counter for each flow.
impl IpIdGeneratorFactory for LinuxIpId {
    fn create(&self, conn: &Connection) -> Box<dyn IpIdGenerator + Send> {
        Box::new(Counter::seeded(hash(conn)))
    }
}

// Windows: one counter for each target, shared by all flows to it.
impl IpIdGeneratorFactory for WindowsIpId {
    fn create(&self, conn: &Connection) -> Box<dyn IpIdGenerator + Send> {
        Box::new(self.per_target.counter_for(conn.remote.addr))
    }
}

// macOS, iOS: no IP ID.
impl IpIdGeneratorFactory for AppleIpId {
    fn create(&self, _conn: &Connection) -> Box<dyn IpIdGenerator + Send> {
        Box::new(Zero)
    }
}
```

Fingerprint and Demo

```
    .-[ 127.0.0.1/42136 -> 127.0.0.1/1234 (syn) ]-
|
| client   = 127.0.0.1/42136
| os       = Windows/7 or 8
| dist     = 0
| params   = Generic
| raw_sig  = 4:128+0:0:65535:mss*4,0:mss,nop,ws,nop,nop,sok:df,id+:0
```

FIN.

JOE DYE



BYTEFUL.COM