HILTI: An Abstract Execution Environment for Deep, Stateful Network Traffic Analysis

Robin Sommer

International Computer Science Institute, &
Lawrence Berkeley National Laboratory

robin@icsi.berkeley.edu
http://www.icir.org/robin

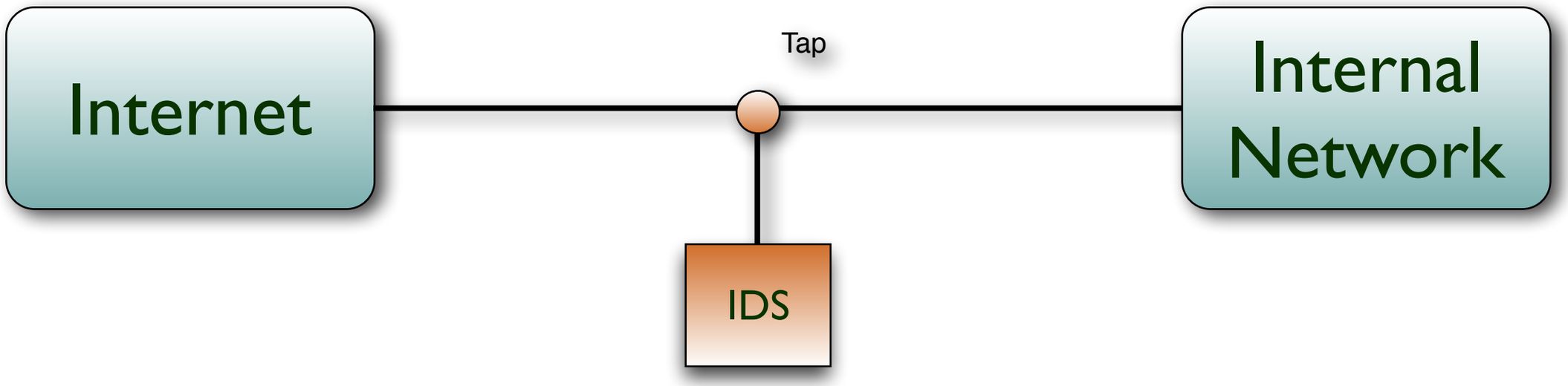
A Tale of Three Open-Source IDS

Shared functionality?

Essentially none.

Same for packet filters, firewalls, proxies, routers, switches, OS stack …

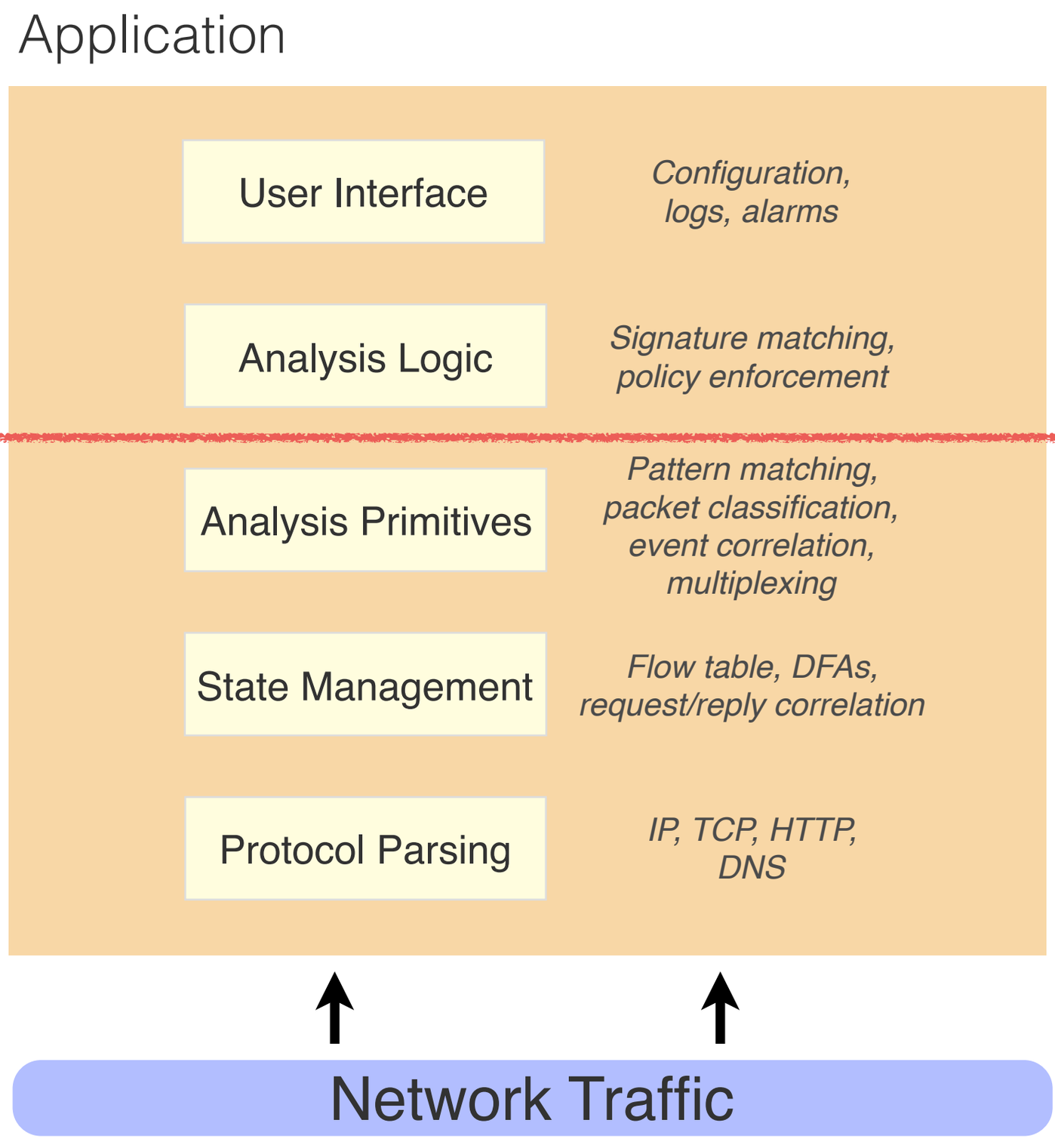
Deep Packet Inspection



Example: Finding downloads of known malware.

1. Find and parse all Web traffic.
2. Find and extract binaries.
3. Compute hash and compare with database.
4. Report, and potentially kill, if found.

DPI Architecture



Common primitives & idioms — but hardly any reuse …

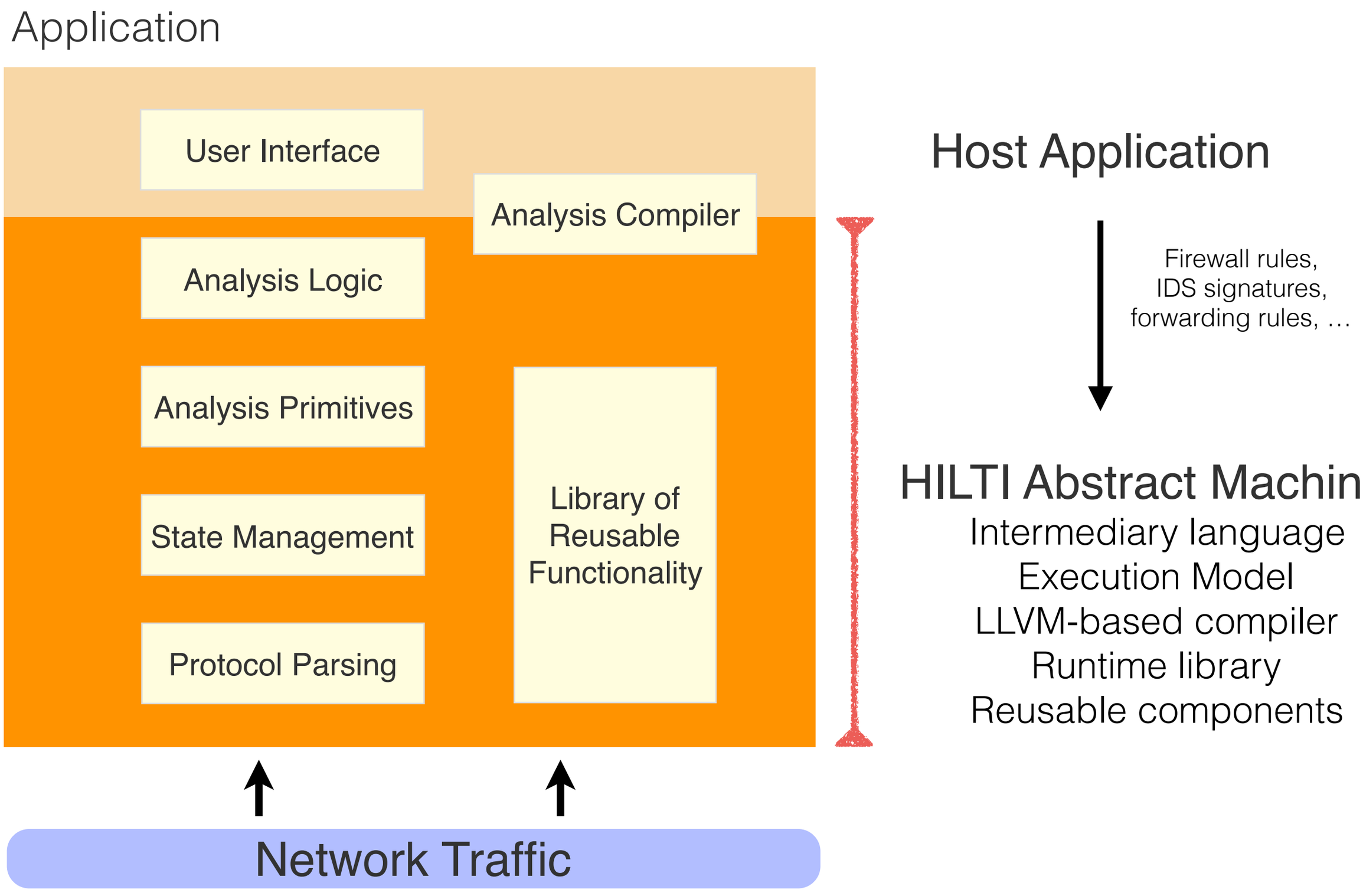
… and that even though this stuff is hard.

Why?

Different low-level structure & data flows.

No “common language”.

A High-Level Intermediary Language for Traffic Inspection



Example: BPF Filters

```
host 192.168.1.1 or src net 10.0.5.0/24
```

```
type IP::Header = overlay {
  hdr_len: int<8> at 0 unpack UInt8InBigEndian (0, 3),
  version: int<8> at 0 unpack UInt8InBigEndian (4, 7),
 [...]
  src:     addr   at 12 unpack IPv4InNetworkOrder,
  dst:     addr   at 16 unpack IPv4InNetworkOrder
}
```

```
bool filter(ref<bytes> packet) {
  local addr a1, a2
  local bool b1, b2, b3

  a1 = overlay.get IP::Header src packet
  b1 = equal a1 192.168.1.1
  a1 = overlay.get IP::Header dst packet
  b2 = equal a2 192.168.1.1
  b1 = or b1 b2
  b2 = equal 10.0.5.0/24 a1
  b3 = or b1 b2
  return b3
}
```

Instruction Set

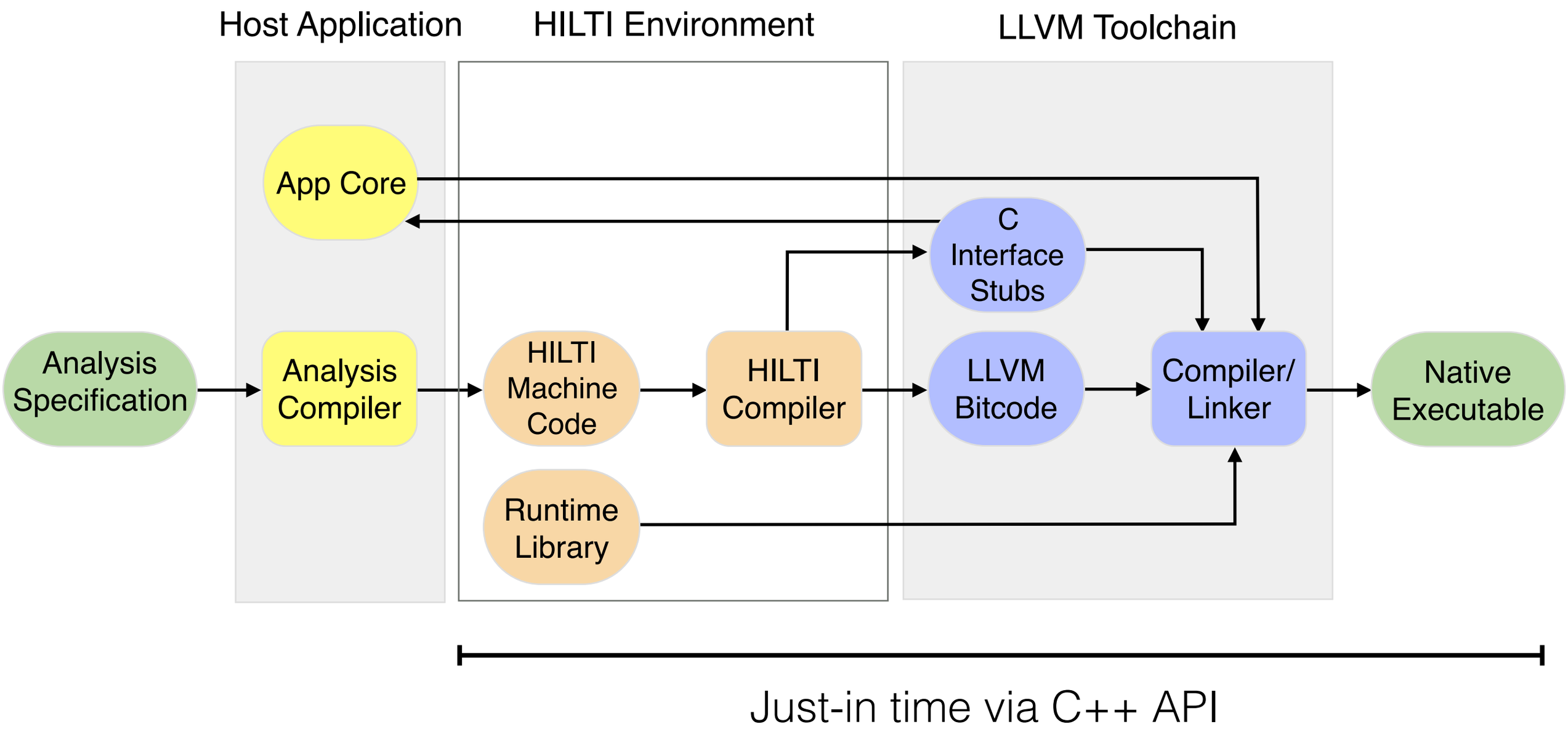
Bitsets	Packet input
Booleans	Packet classification
CIDR masks	Packet dissection
Callbacks	Ports
Closures	Profiling
Channels	Raw data
Debug support	References
Doubles	Regular expressions
Enumerations	Strings
Exceptions	Structs
File i/o	Unions
Flow control	Time intervals
Hashmaps	Timer management
Hashsets	Timers
IP addresses	Times
Integers	Tuples
Lists	Vectors/arrays

HILTI Machine Model

Focus Areas

- Rich Domain-specific Data Types
- Flexible Control Flow
- Concurrent Analysis
- Robust & Secure Execution
- Comprehensive Host Interface
- Real-time Performance
- Debugging & Profiling Support
- High-level Optimization

Implementation: The HILTI Toolchain



Hello, World!

```
module Main

import Hilti

void run() {
    call Hilti::print("Hello, World!")
}
```

hello.hlt

```
# hilti-build hello.hlt -o a.out && ./a.out
Hello, World!
```

```
# hiltic -j hello.hlt
Hello, World!
```

Can HILTI support complex applications?

Application Case Studies

BPF Filter

Stateful Firewall

Protocol Parsing

Bro Script Execution

BinPAC - A Yacc for Network Protocols

Grammar example: Parsing SSH banners.

`SSH-2.0-OpenSSH_3.8.1p1`

```
type SSH::Banner = unit {
    magic   : /SSH-/;
    version : /[^-]*/;
    dash    : /-/;
    software: /[^\r\n]*/;
}
```

BinPAC compiles grammar into HILTI parser.

HILTI compiles parser into executable code just-in-time.

Bro plugin integrates parsers at startup.

Hello, World!

```
type SSH::Banner = unit {
    magic   : /SSH-/;
    version : /[^-]*/;
    dash    : /-/;
    software: /[^\r\n]*/;
}
```

ssh.pac2

```
grammar ssh.pac2;

protocol analyzer SSH over TCP:
parse with SSH::Banner, port 22/tcp;

on SSH::Banner
    -> event ssh_banner(self.version, self.software);
```

ssh.evt

```
event ssh_banner(version: string, software: string) {
    { print software, version; }
```

ssh.bro

```
# bro -r ssh.trace ssh.evt ssh.bro
OpenSSH_3.9p1, 1.99
OpenSSH_3.8.1p1, 2.0
```

Application Case Studies

BPF Filter

Stateful Firewall

Protocol Parsing

Bro Script Execution

Bro Scripts

Script example: A simple scan detector.

```
global attempts: table[addr] of count &default=0;

event connection_rejected(c: connection)
{
    local orig = c$id$orig_h;   # Get originator address.

    local n = ++attempts[orig]; # Increase counter.

    if ( n == SOME_THRESHOLD )  # Check for threshold.
        NOTICE(...);            # Alarm.
}
```

Bro plugin compiles scripts into HILTI code.

HILTI compiles that into executable code just-in-time.

Evaluation

Use HILTI plugin for Bro to compare parsing & script execution with a native Bro.

Traces: HTTP: 1/25 of Berkeley port 80 traffic.
30GB trace, 52min, 340k messages.

DNS: Full Berkeley port 53 traffic.
1GB trace, 10min, 65M messages.

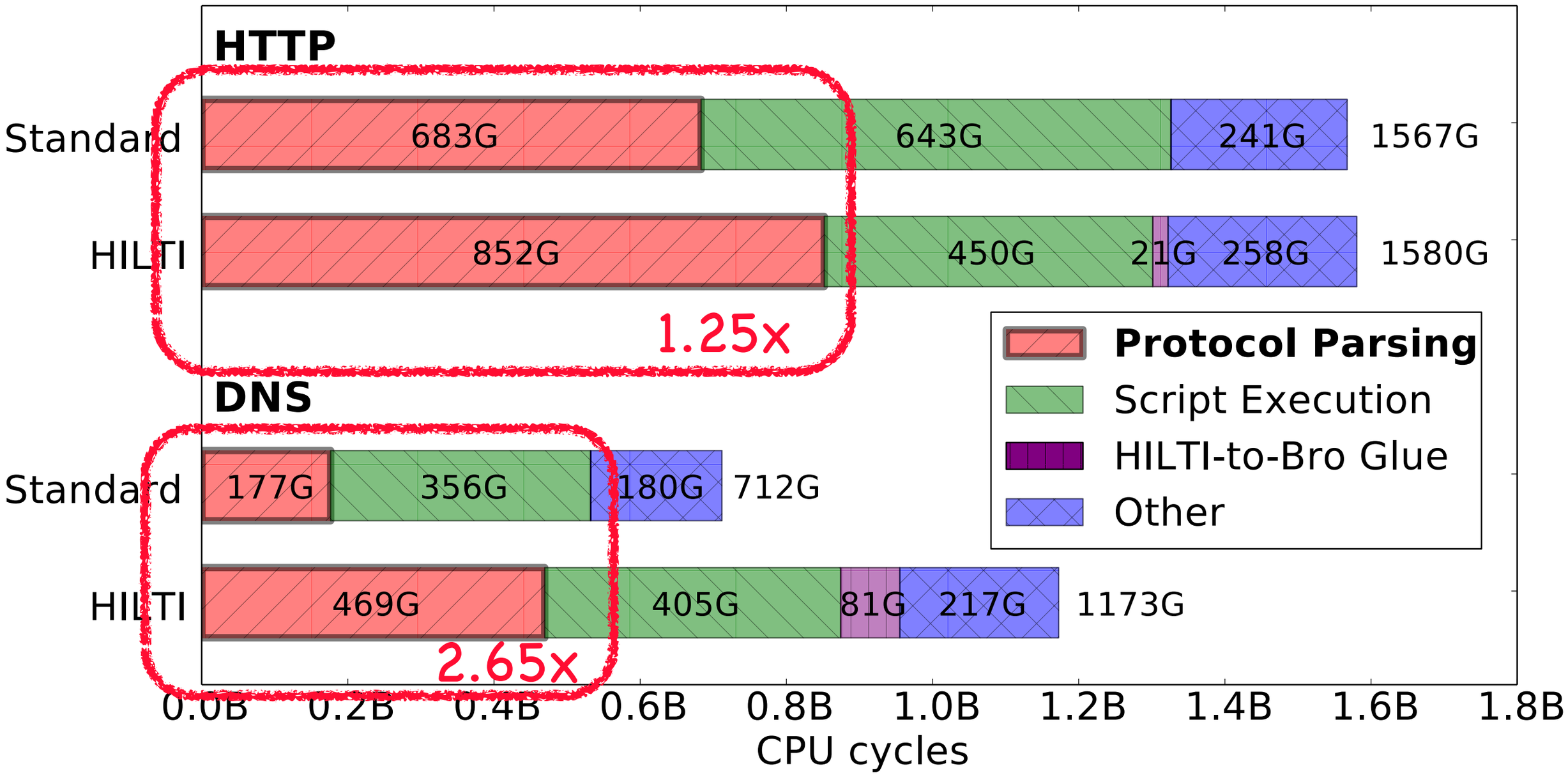
Correctness

HILTI captures semantics correctly.

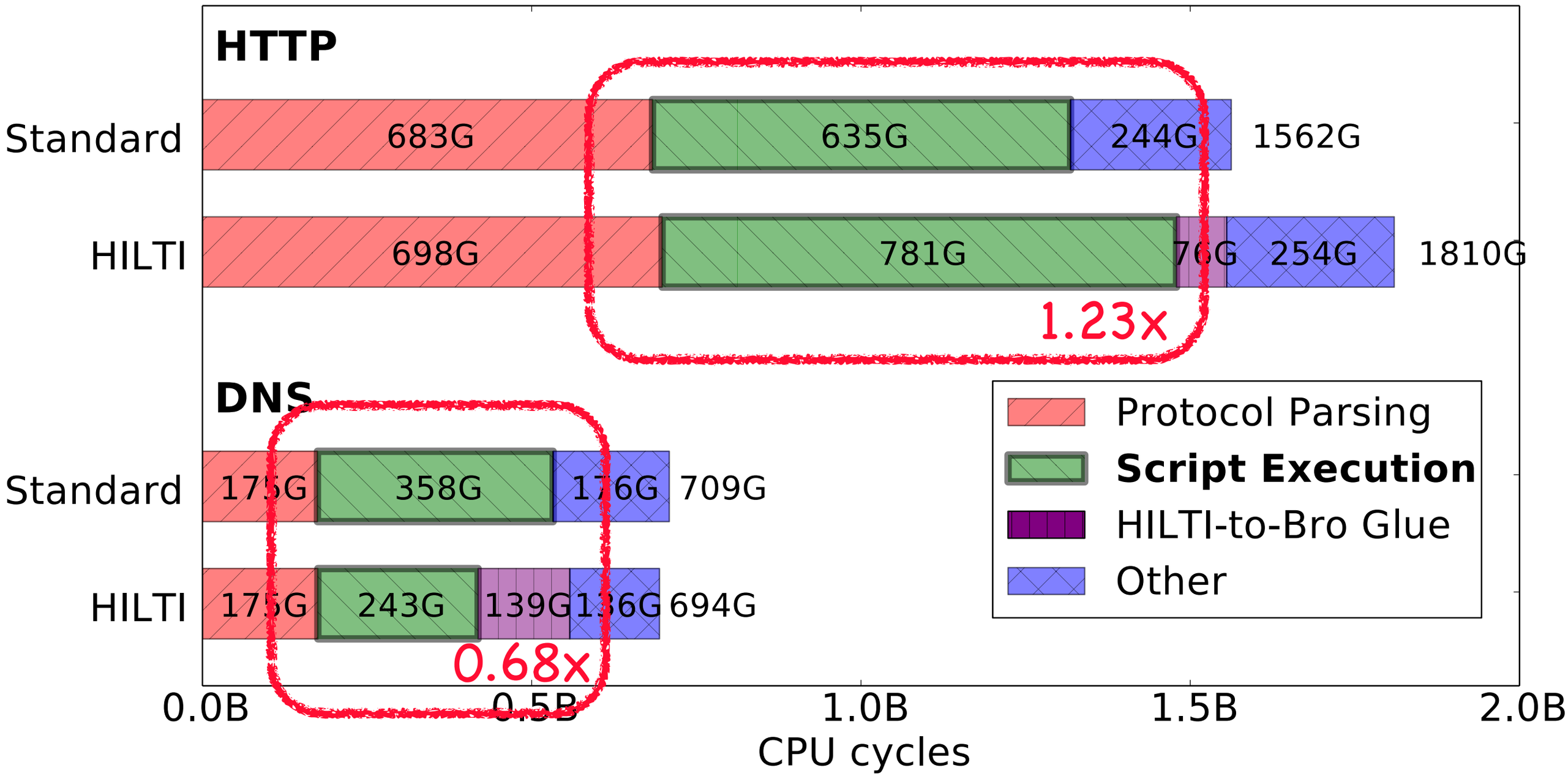
Performance

Let's see.

Protocol Parsing



Bro Scripts


HTTP
Standard
683G
635G
244G
1562G
HILTI
698G
781G
76G
254G
1810G
1.23x
DNS
Standard
175G
358G
176G
709G
HILTI
175G
243G
139G
136G
694G
0.68x
Protocol Parsing
Script Execution
HILTI-to-Bro Glue
Other
0.0B
0.5B
1.0B
1.5B
2.0B
CPU cycles

Summary

HILTI: A new platform for network traffic analysis.

- A compiler-target for host applications to leverage.
- Provides common data structures and control flow primitives.

Case studies demonstrate aptness of design.

- Packet filter, stateful firewall, protocol parsing, Bro scripts.

Initial performance experiments encouraging.

- Not too different from native applications.
- It's still a prototype, with lots of potential.

Sommer/Vallentin/De Carli/Paxson: "HILTI: An Abstract Execution Environment for Deep, Stateful Network Traffic Analysis". ACM IMC 2014.

The HILTI Vision

Performance via Abstraction

Transparent improvement under the hood.
Integration of non-standard hardware.
High-level compiler optimizations.
Automatic parallelization.*

Facilitate Reuse

Means and glue to share functionality.
HILTI library of common high-level components.

HILTI is available under BSD license at

`http://www.icir.org/hilti`

* De Carli/Sommer/Jha: “Beyond Pattern Matching: A Concurrency Model for Stateful Deep Packet Inspection”. ACM CCS 2014.