Iris: Expressive Traffic Analysis for the Modern Internet

Thea Rossman, Diana Qing, Gerry Wan, Zakir Durumeric
Stanford University

Empirical Security Research Group

Network Traffic Analysis is Critical

Strengthen security and privacy

Improve performance and QoS

Build and evaluate new systems

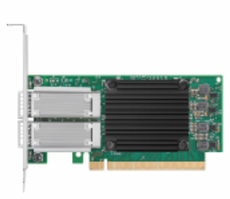
Measure Internet phenomena

Demands on Traffic Analysis Systems are Increasing

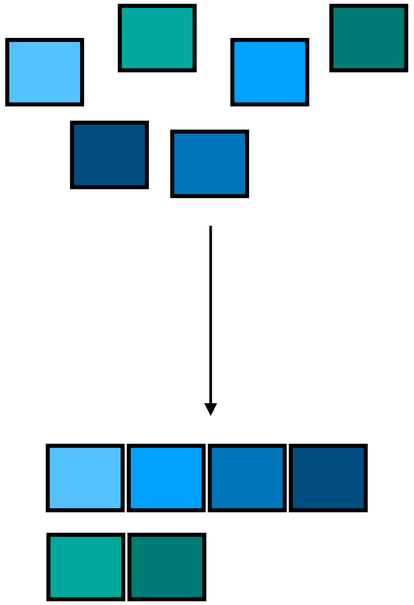
Demands on Traffic Analysis Systems are Increasing

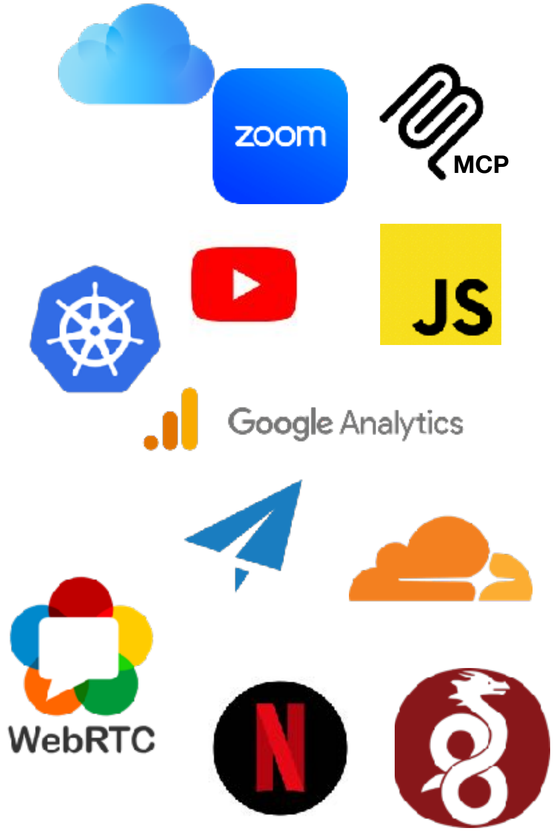
High Traffic Volumes (100Gbps+)

Demands on Traffic Analysis Systems are Increasing



High Traffic Volumes (100Gbps+)





Complex Applications and Questions

Two Choices Today

Option 1
Build an experiment-specific system from scratch

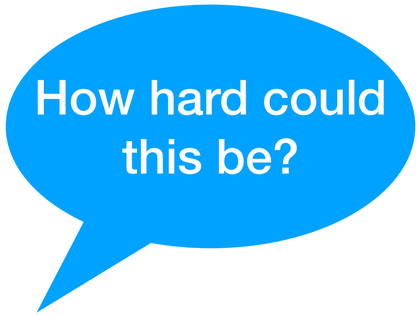
AF_PACKET

Option 2
Try to use an existing tool, which may not meet needs

Pigasus

Traffic Refinery

Two Choices Today



Option 1
Build an experiment-specific system from scratch

Option 2
Try to use an existing tool, which may not meet needs

Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

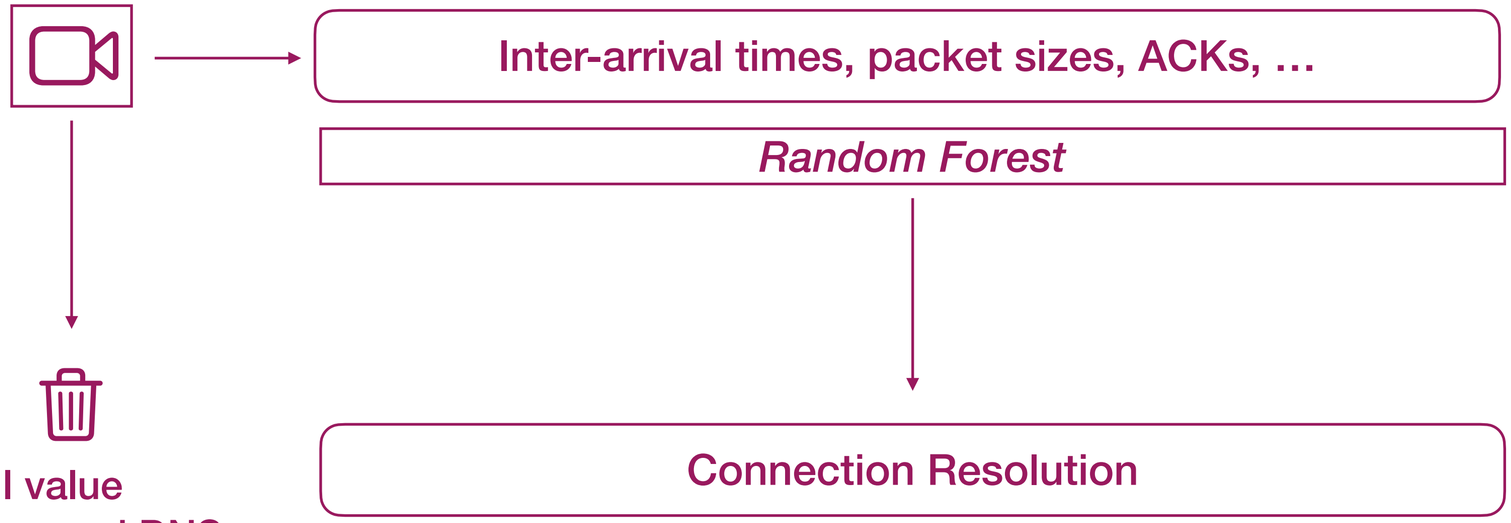
“What is the distribution of video resolution on my network?”

Based on:
F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. SIGMETRICS 2019.

Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

“What is the distribution of video resolution on my network?”

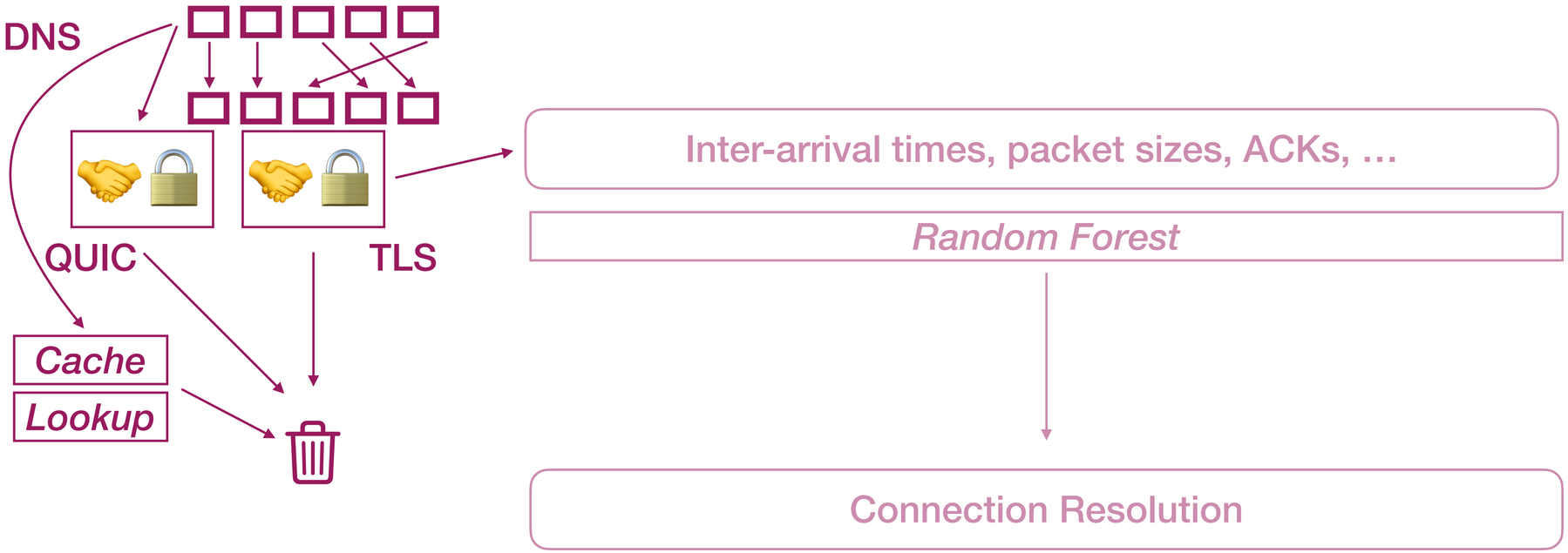


Based on:
F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. SIGMETRICS 2019.

Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

“What is the distribution of video resolution on my network?”

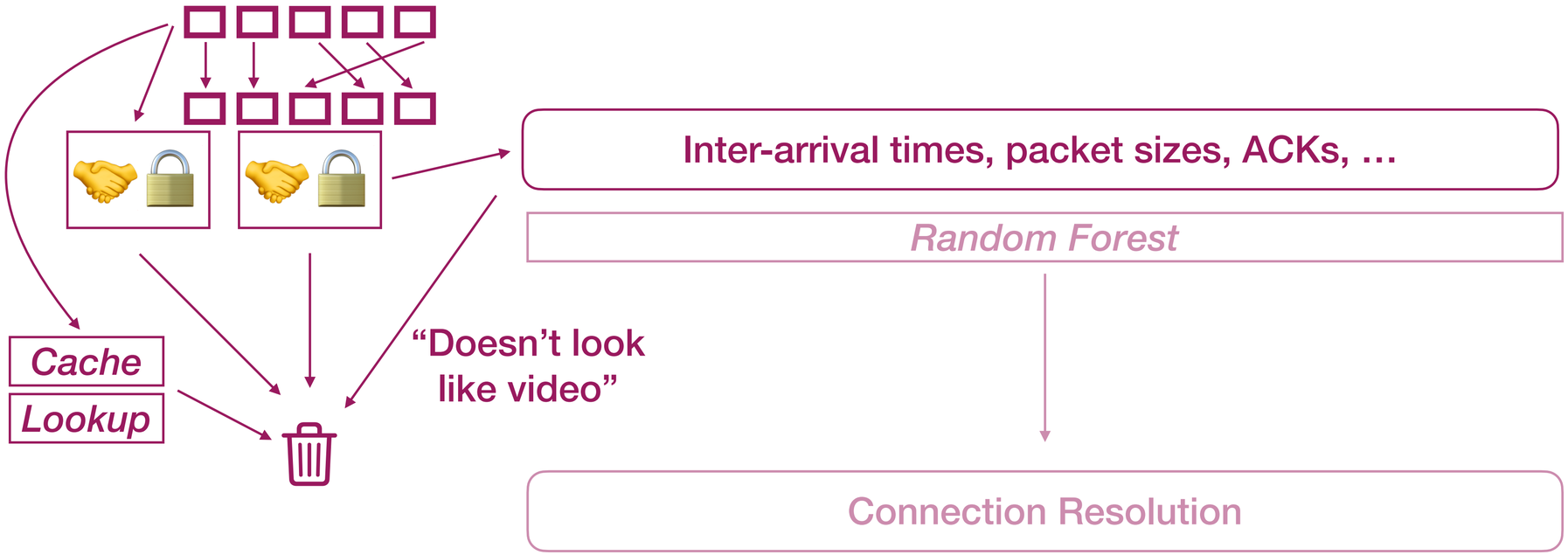


Based on:
F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. SIGMETRICS 2019.

Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

“What is the distribution of video resolution on my network?”

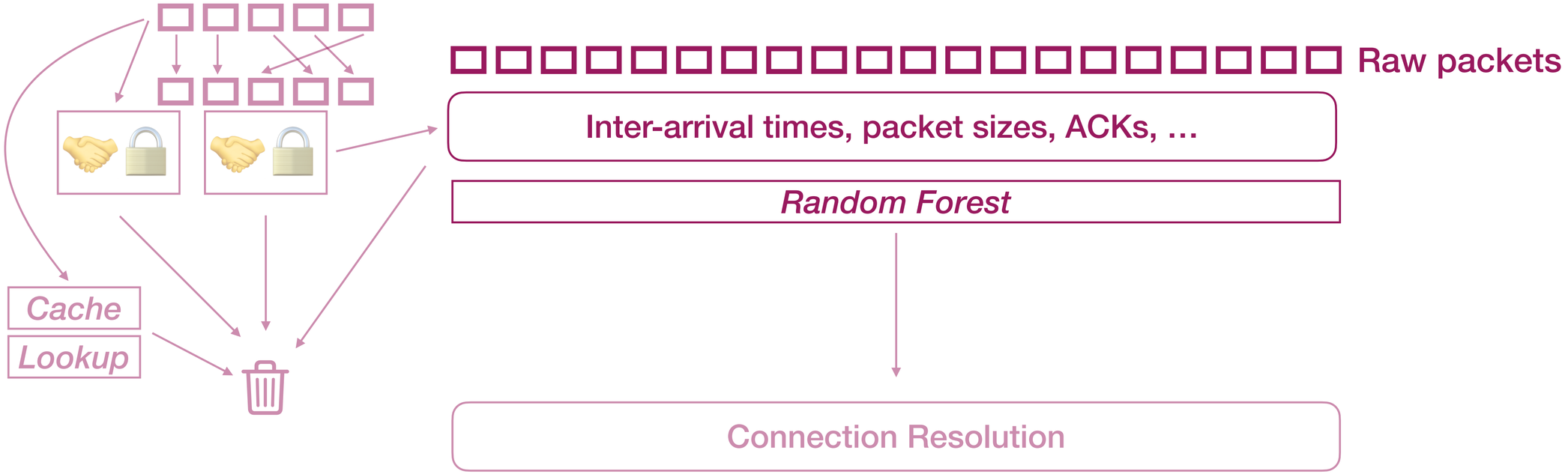


Based on:
F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. SIGMETRICS 2019.

Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

“What is the distribution of video resolution on my network?”

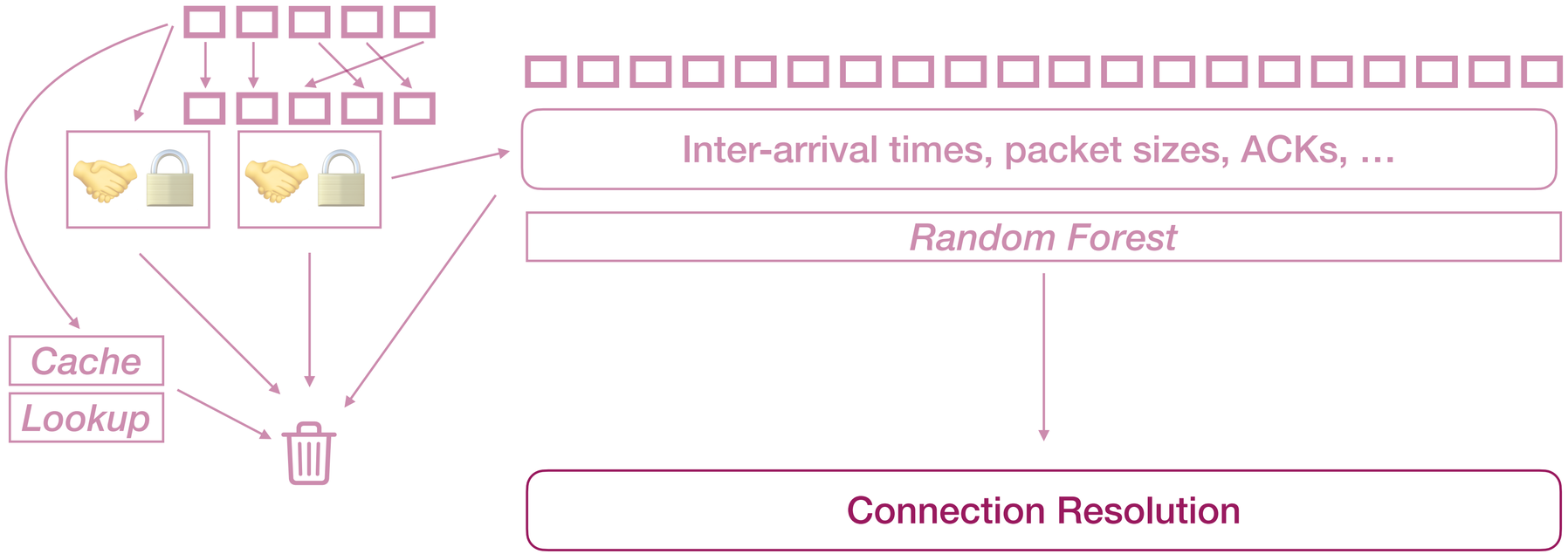


Based on:
F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. SIGMETRICS 2019.

Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

“What is the distribution of video resolution on my network?”

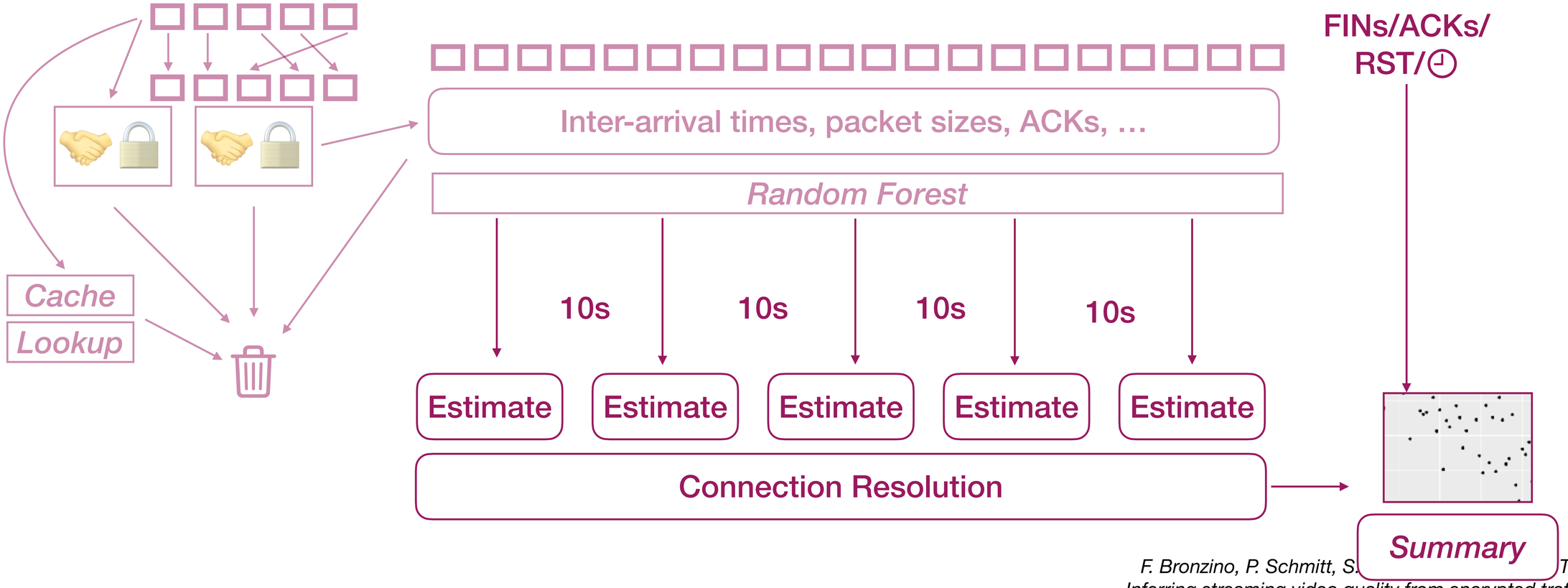


Based on:
F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. SIGMETRICS 2019.

Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

"What is the distribution of video resolution on my network?"

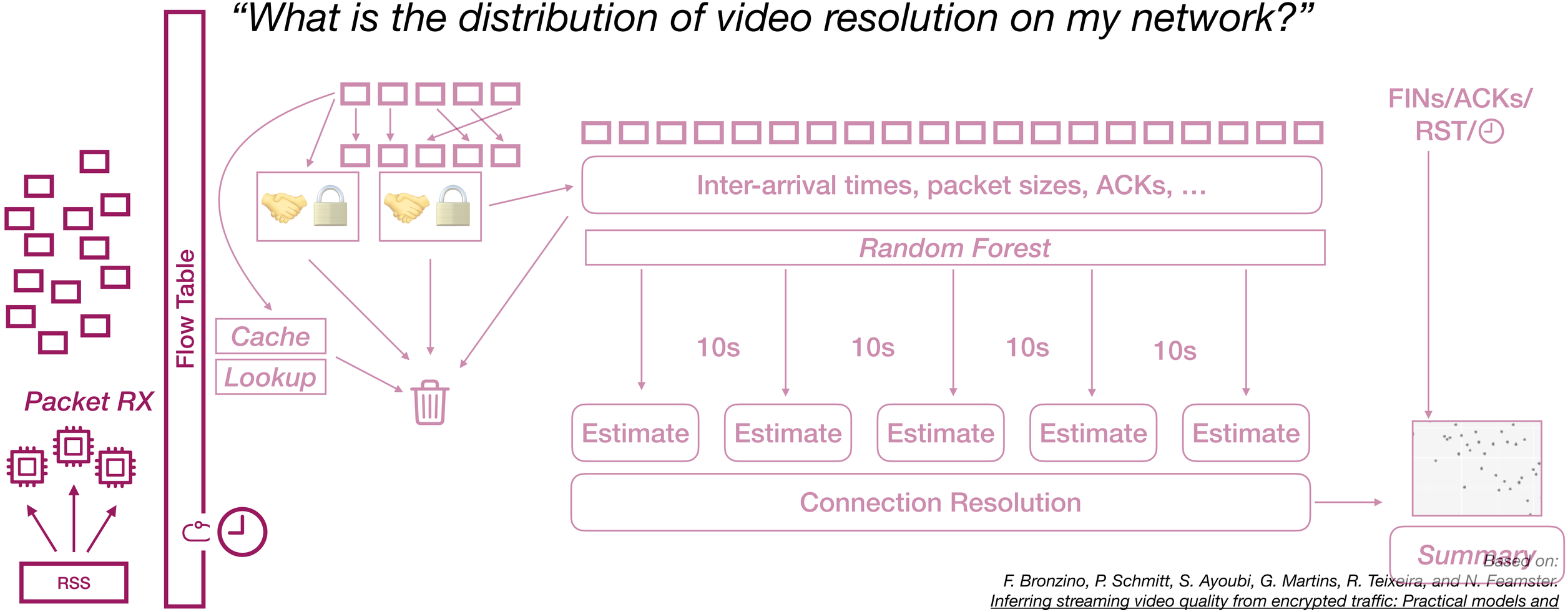


Based on:
F. Bronzino, P. Schmitt, S. [illegible] Teixeira, and N. Feamster. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. SIGMETRICS 2019.

Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

"What is the distribution of video resolution on my network?"



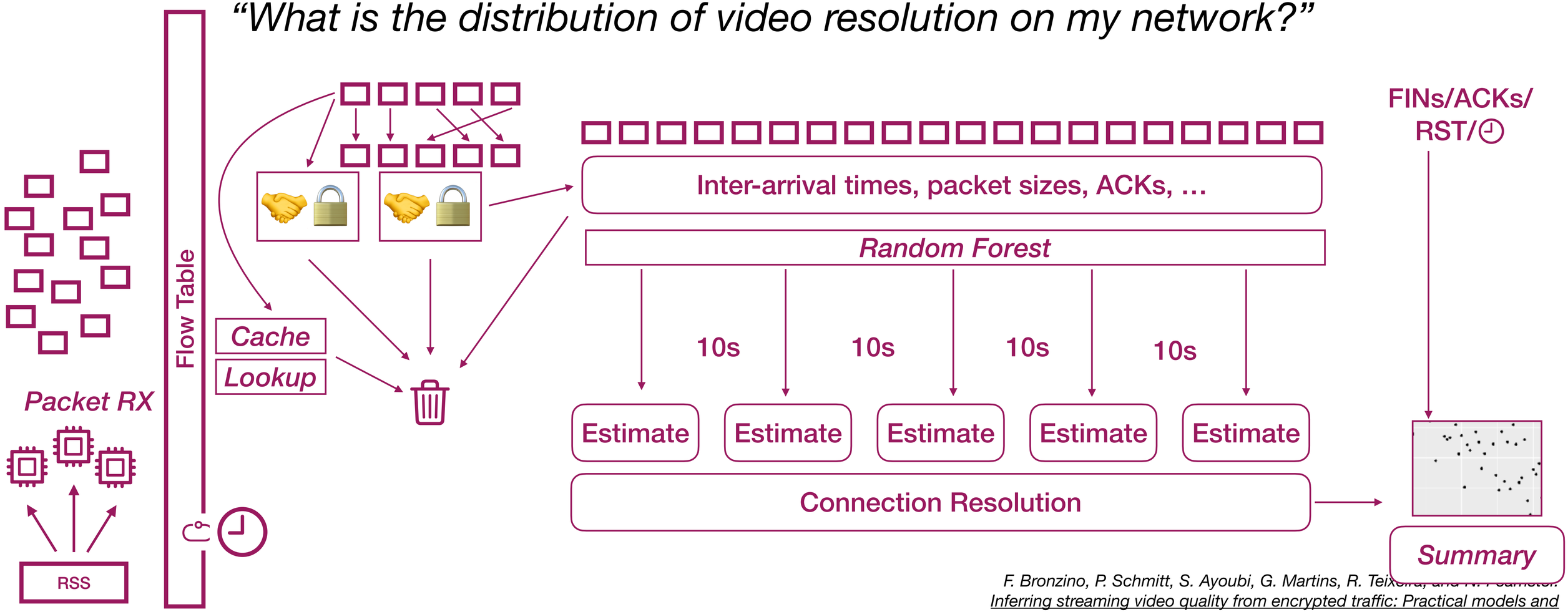
Based on:

F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. SIGMETRICS 2019.

Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

"What is the distribution of video resolution on my network?"

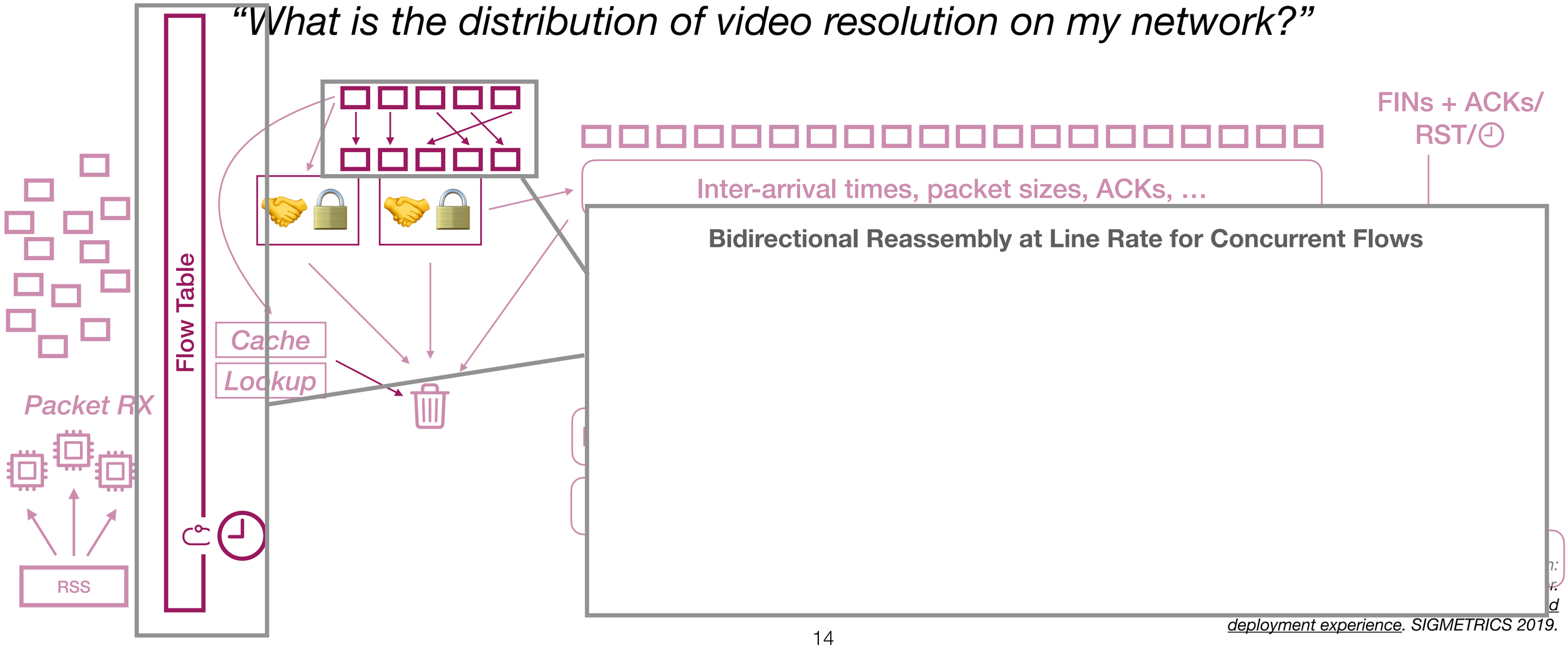


F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. SIGMETRICS 2019.

Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

"What is the distribution of video resolution on my network?"

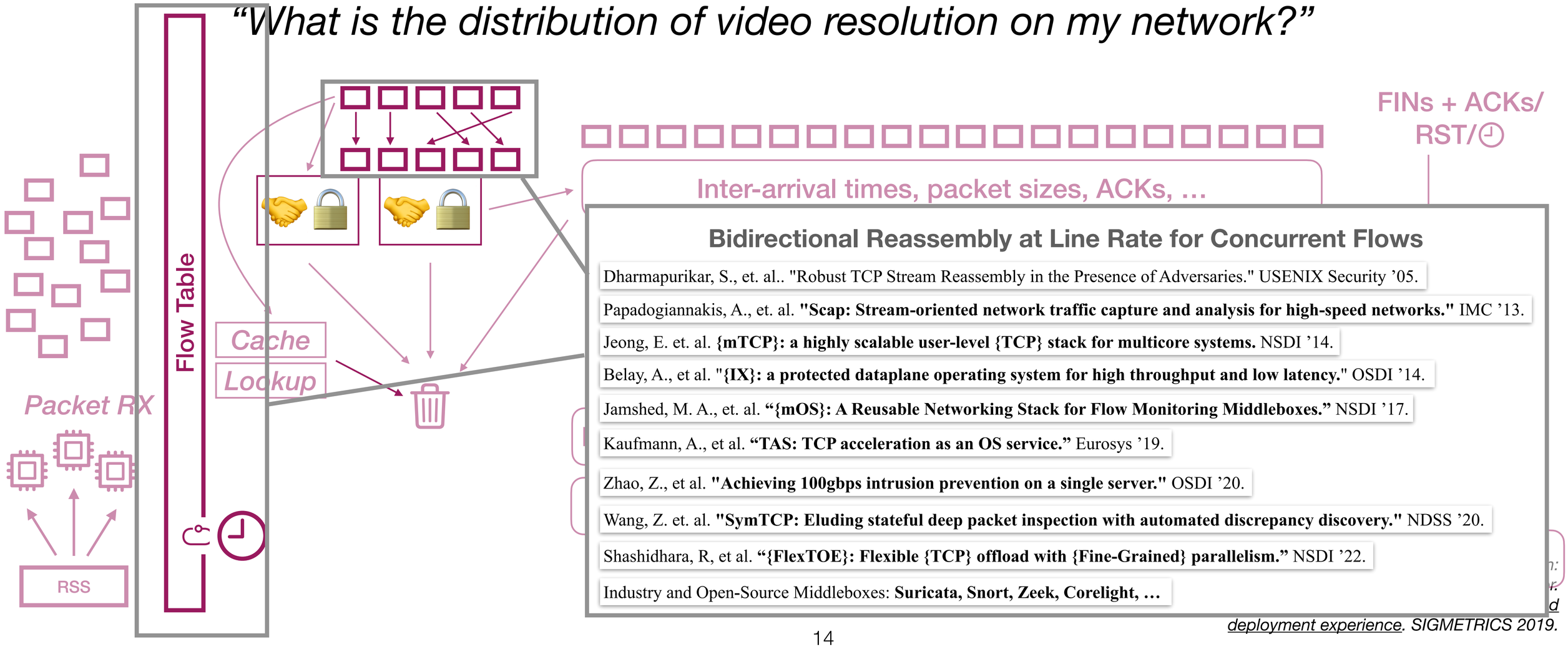


deployment experience. SIGMETRICS 2019.

Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

"What is the distribution of video resolution on my network?"



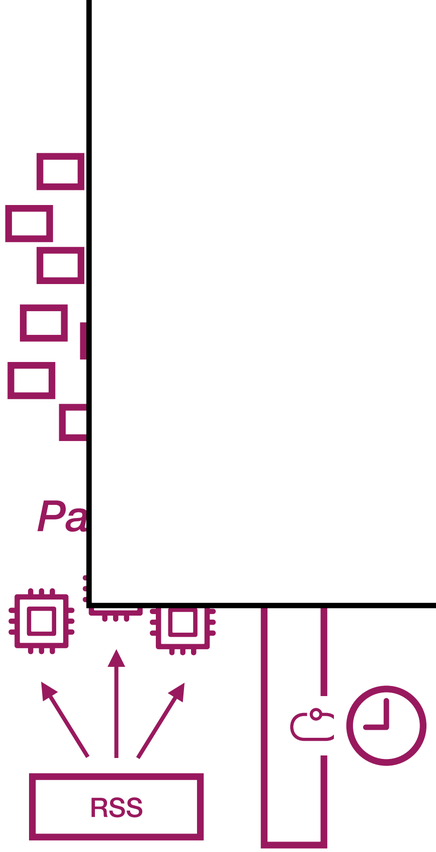
deployment experience. SIGMETRICS 2019.

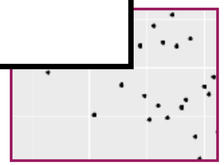
Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

- *“What is the distribution of video resolution on my network?”*

Operators and researchers typically need to run multiple experiments in parallel.





Summary

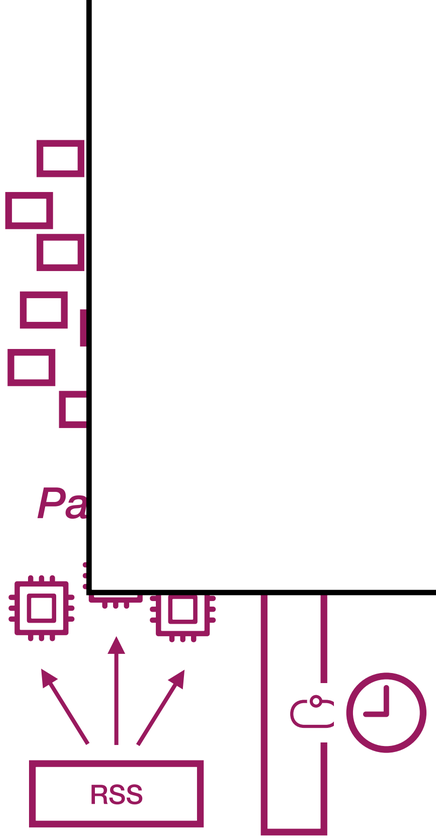
F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. SIGMETRICS 2019.

Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

- *“What is the distribution of video resolution on my network?”*

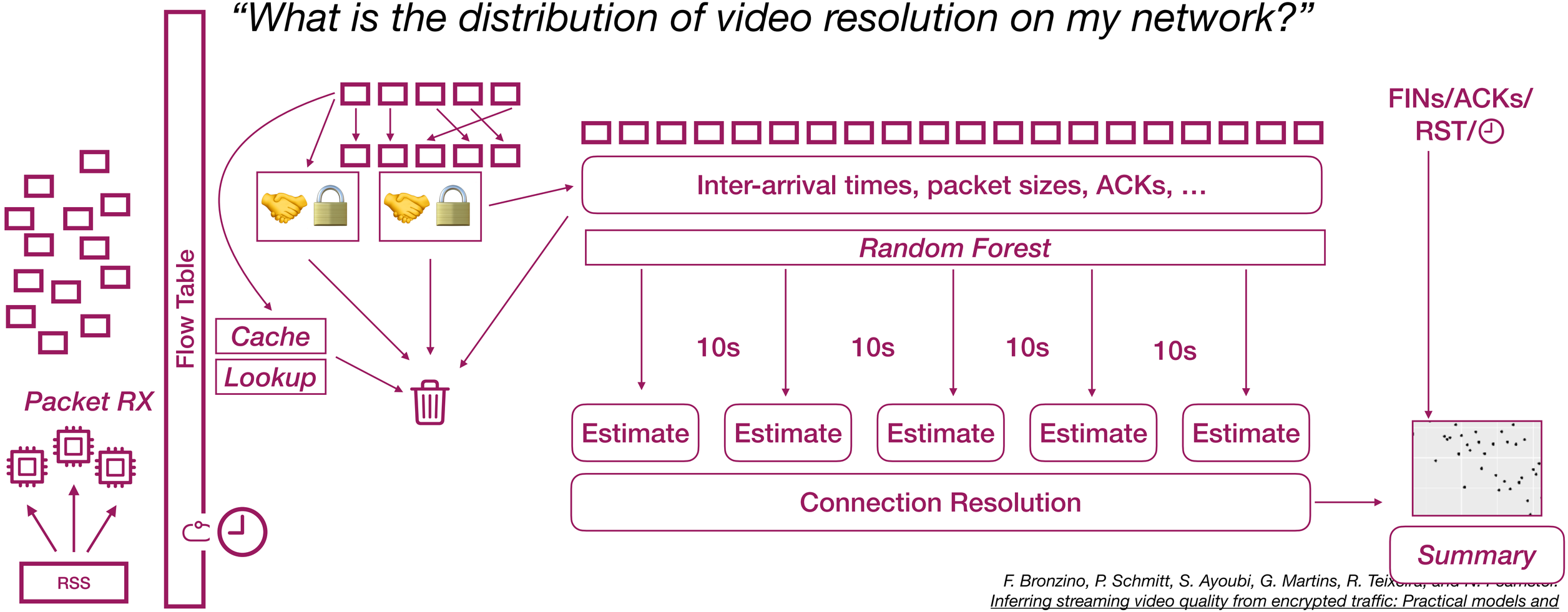
Operators and researchers typically need to run multiple experiments in parallel.



Motivating Example: Video Resolution

Seemingly simple questions require **thousands of lines of code** to implement.

"What is the distribution of video resolution on my network?"



F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster. Inferring streaming video quality from encrypted traffic: Practical models and deployment experience. SIGMETRICS 2019.

Two Choices Today

Option 1
Build an experiment-specific system from scratch

Option 2
Try to use an existing tool, which may not meet needs

What Do We Need from a Traffic Analysis Tool?

Based on a literature review of 71 recent traffic analysis papers:

What Do We Need from a Traffic Analysis Tool?

Based on a literature review of 71 recent traffic analysis papers:

1. Flexibility

Total programmability across data extraction, filtering, and analysis

What Do We Need from a Traffic Analysis Tool?

Based on a literature review of 71 recent traffic analysis papers:

1. Flexibility

Total programmability across data extraction, filtering, and analysis

2. Abstractions

Access to common, connection- and application-layer types

What Do We Need from a Traffic Analysis Tool?

Based on a literature review of 71 recent traffic analysis papers:

1. Flexibility

Total programmability across data extraction, filtering, and analysis

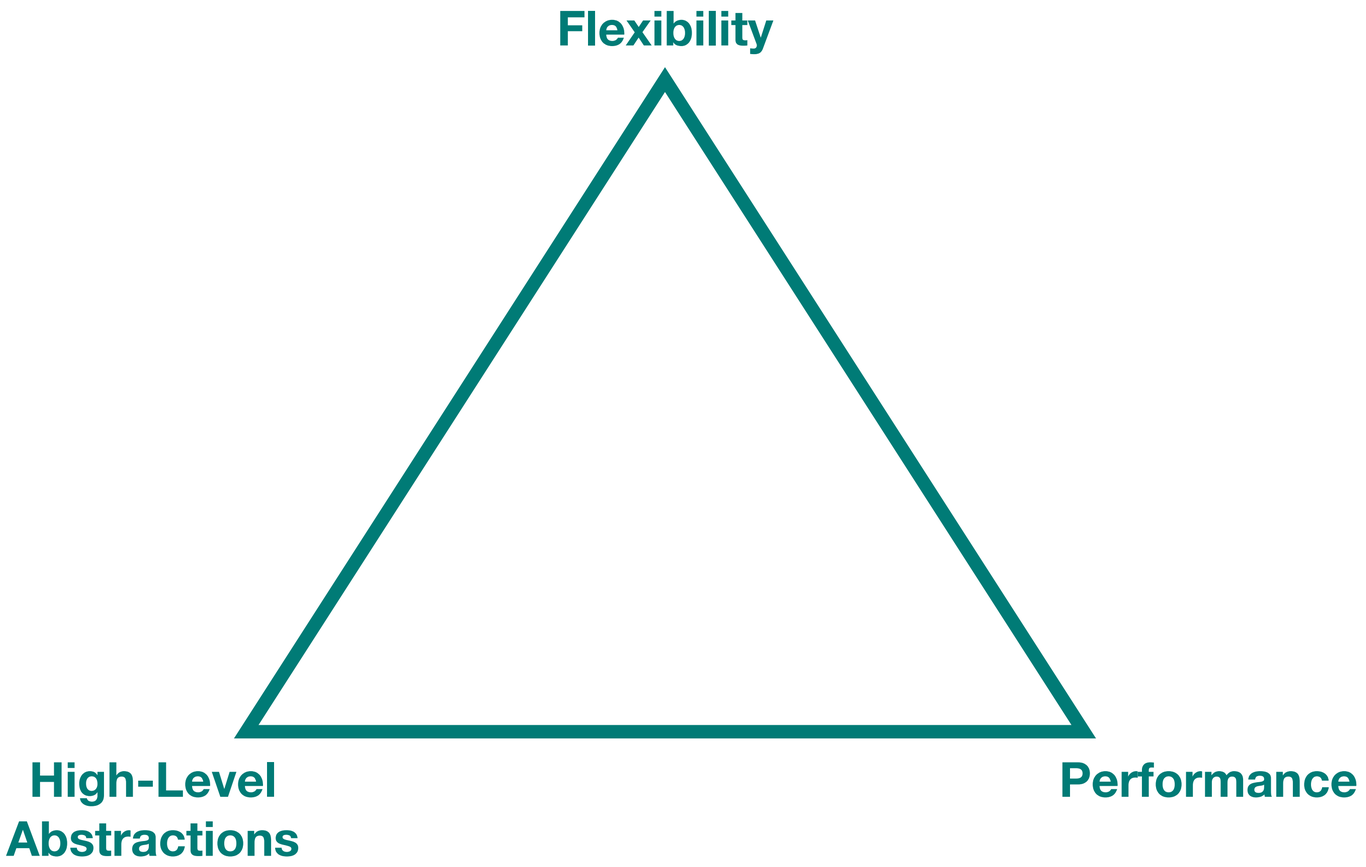
2. Abstractions

Access to common, connection- and application-layer types

3. Performance

100Gbps+ on commodity hardware for concurrent analysis tasks

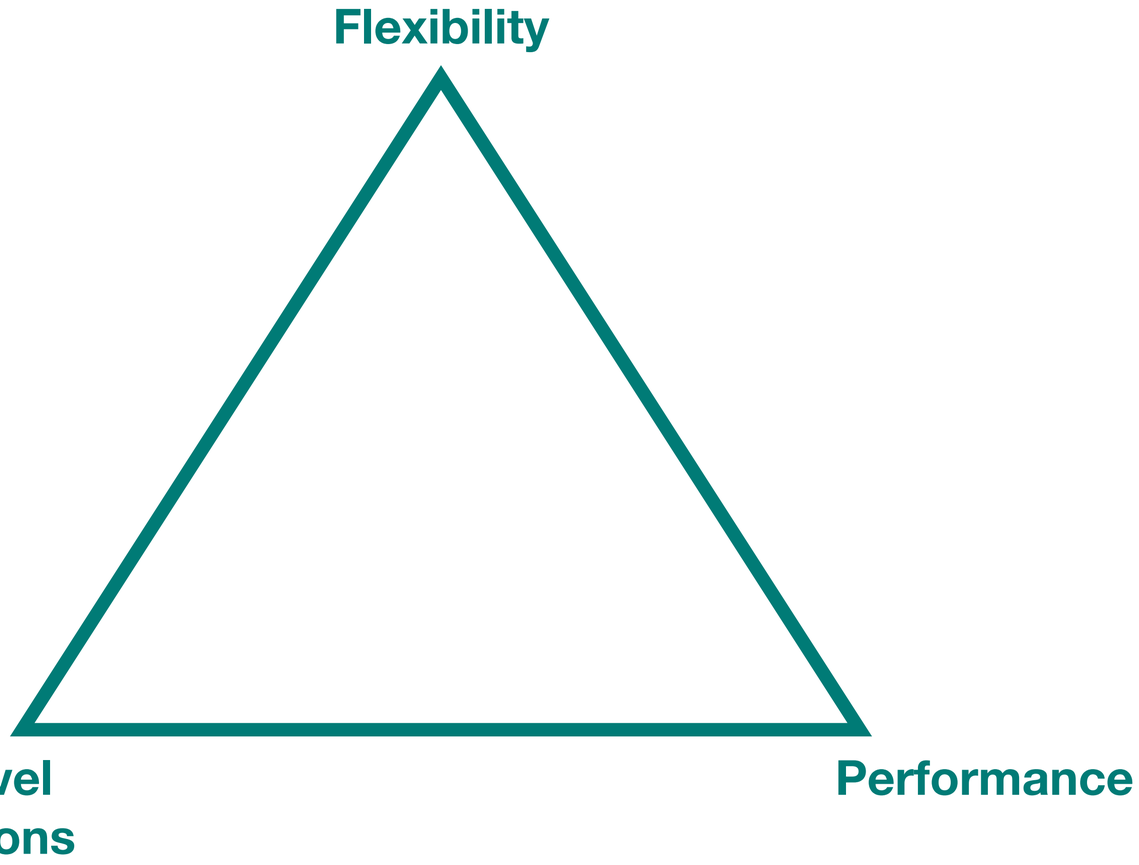
No Current System Provides all Three



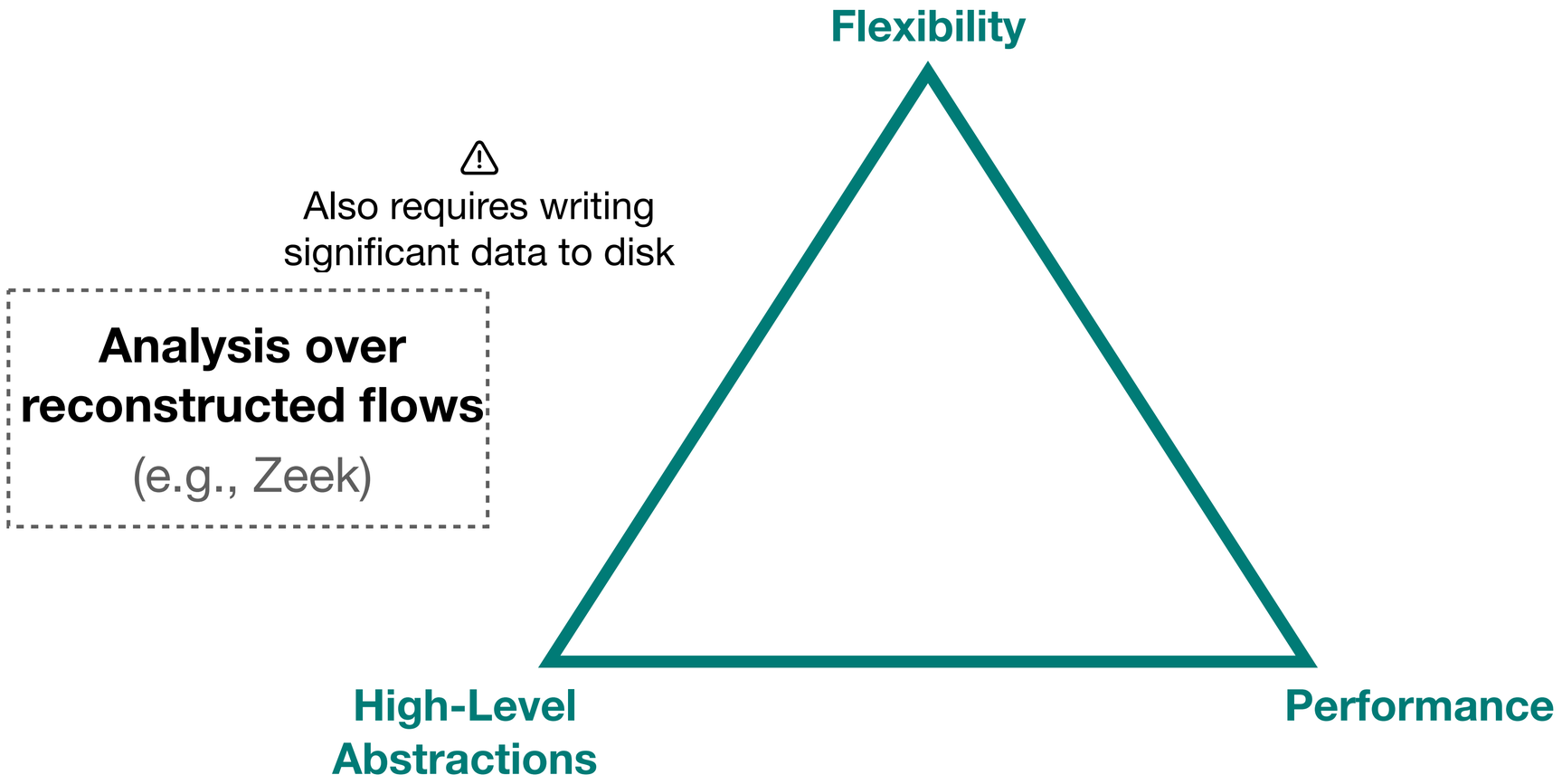
No Current System Provides all Three

Analysis over reconstructed flows

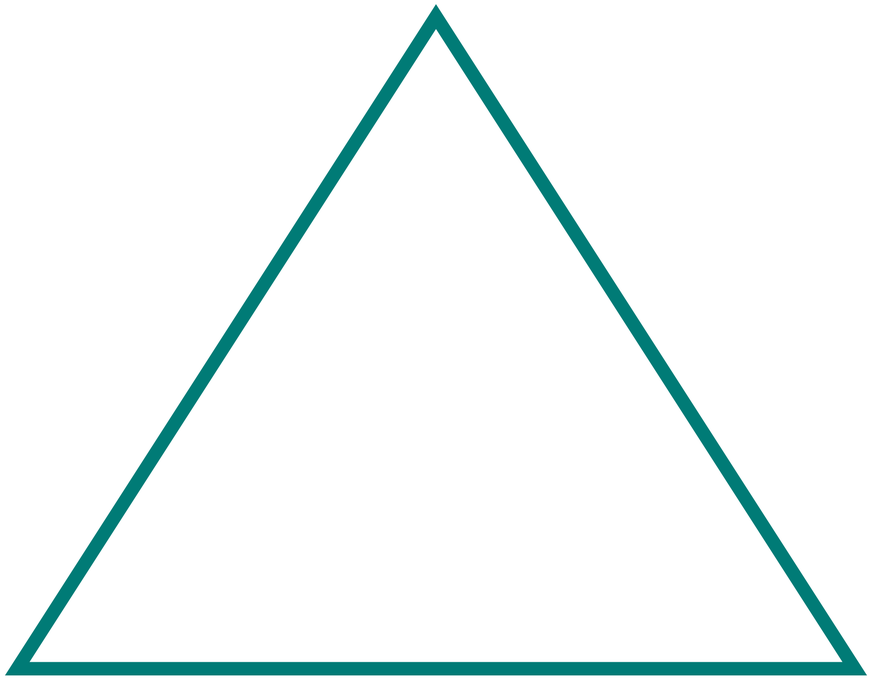
(e.g., Zeek)



No Current System Provides all Three



No Current System Provides all Three



Analysis over reconstructed flows

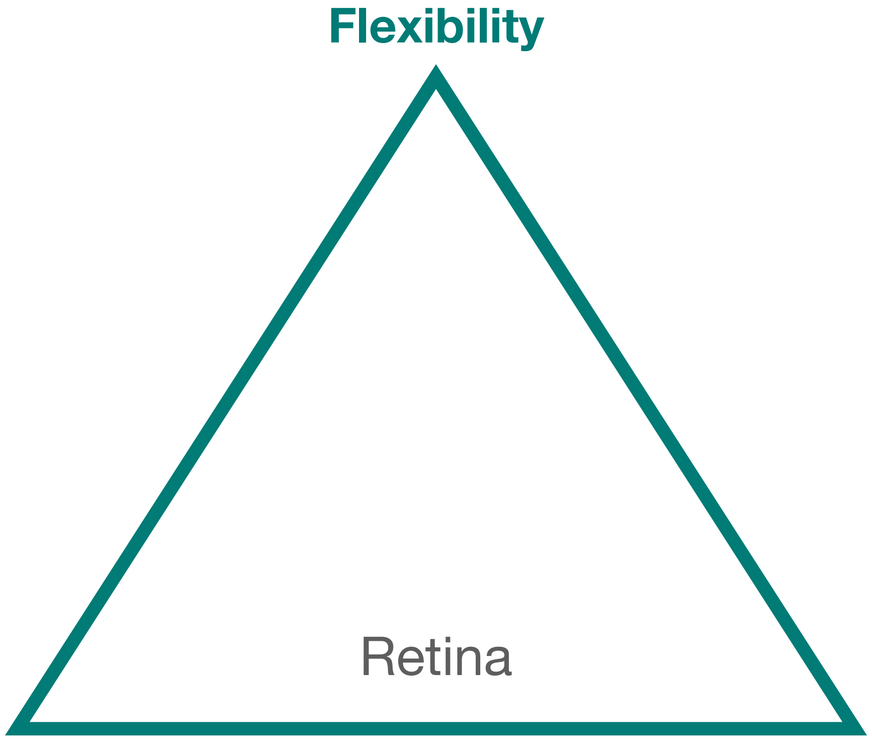
(e.g., Zeek)

Parsing, but limited analysis

(e.g., Pigasus, Traffic Refinery)

NetFlow, Packet DSLs (e.g., Marple, Sonata)

No Current System Provides all Three



Analysis over reconstructed flows

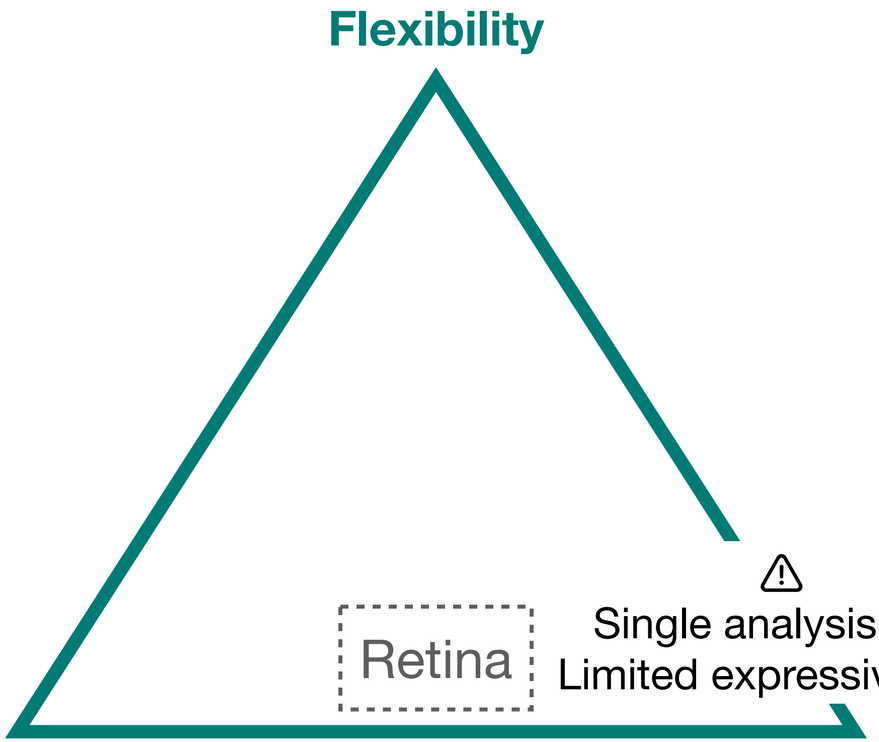
(e.g., Zeek)

Parsing, but limited analysis

(e.g., Pigasus, Traffic Refinery)

NetFlow, Packet DSLs (e.g., Marple, Sonata)

No Current System Provides all Three



Analysis over reconstructed flows

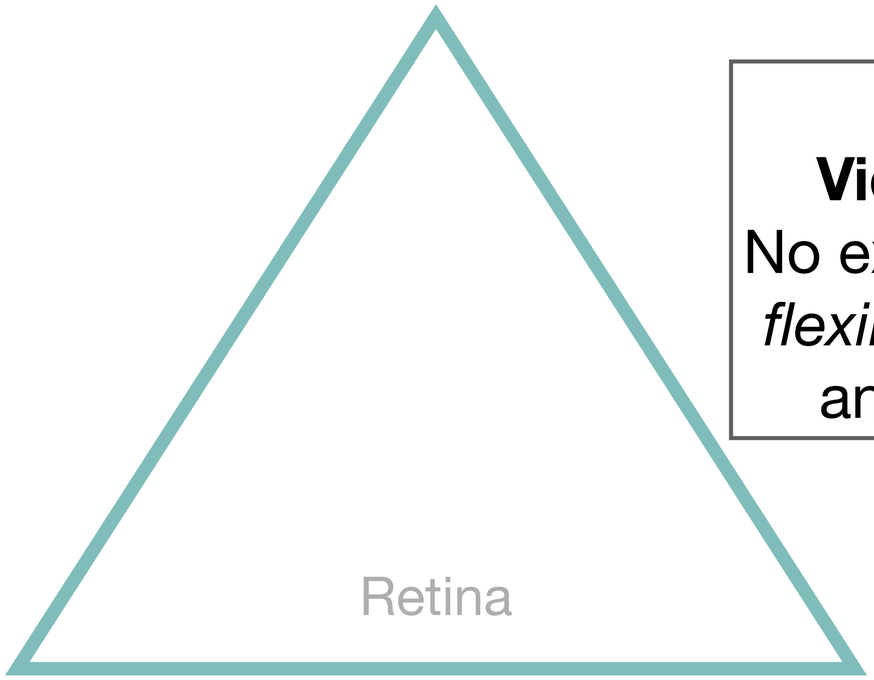
(e.g., Zeek)

Parsing, but limited analysis

(e.g., Pigasus, Traffic Refinery)

NetFlow, Packet DSLs (e.g., Marple, Sonata)

No Current System Provides all Three



Analysis over reconstructed flows

(e.g., Zeek)

Parsing, but limited analysis

(e.g., Pigasus, Traffic Refinery)

NetFlow, Packet DSLs (e.g., Marple, Sonata)

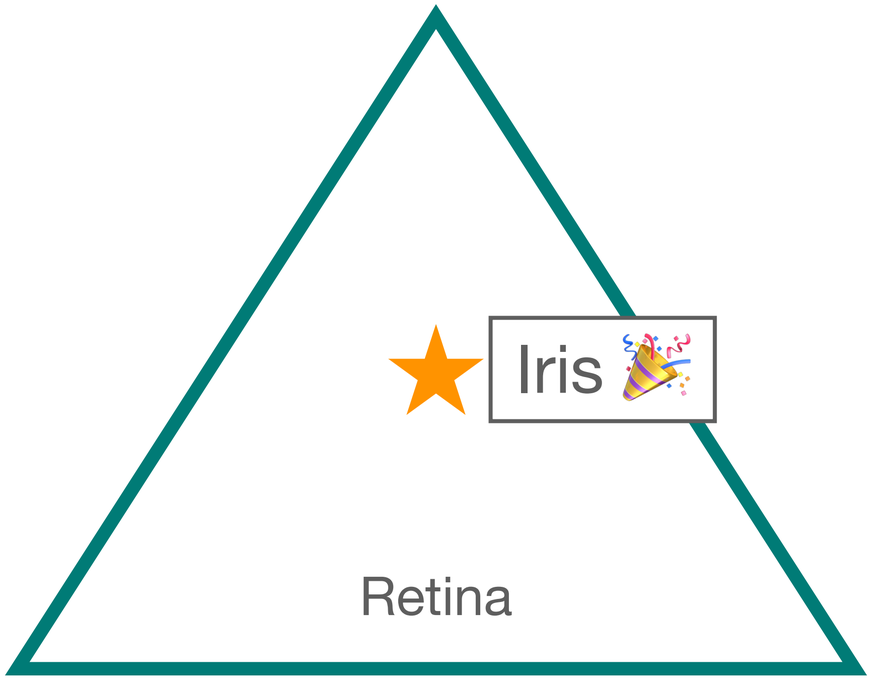
⚠

Video resolution example:
No existing tool provides enough *flexibility*, the right *abstractions*, and sufficient *performance.*

*without creating a new domain-specific tool

No Current System Provides all Three

… Yet?



Analysis over reconstructed flows

(e.g., Zeek)

Parsing, but limited analysis

(e.g., Pigasus, Traffic Refinery)

NetFlow, Packet DSLs (e.g., Marple, Sonata)

Iris

Compile-Time Optimization Enables
High-Performance, Expressive Traffic Analysis

Iris

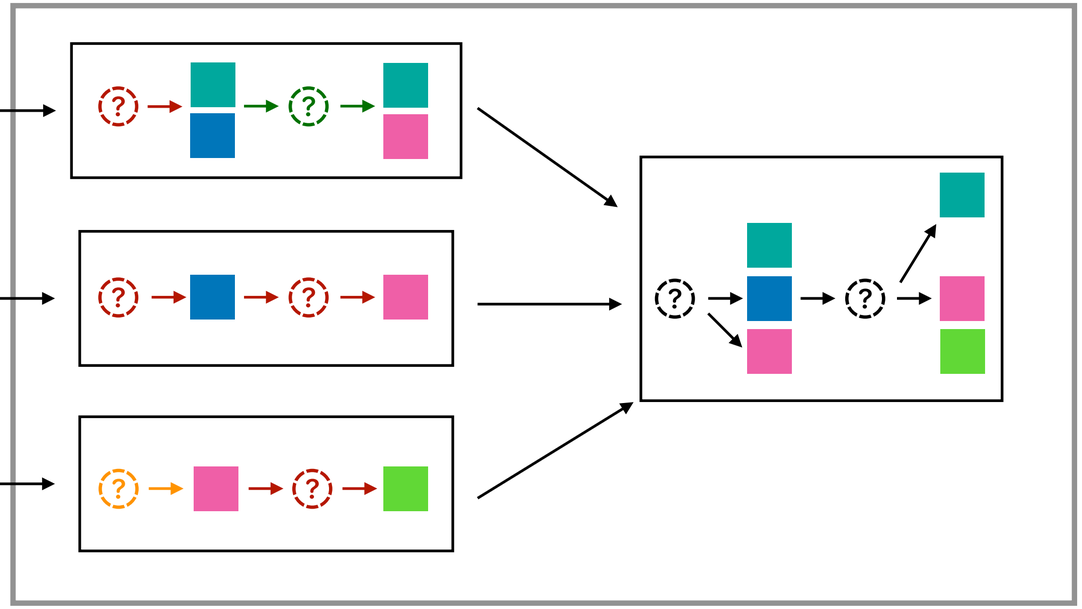
Compile-Time Optimization Enables High-Performance, Expressive Traffic Analysis

```rust
fn video_lookup(
    &mut self, tls: &TlsHandshake,
    ft: &FiveTuple
) -> FilterResult {
    /* ... */
}
```

```rust
#[datatype]
struct FeatureChunk { /* ... */ }

impl FeatureChunk {
   fn new(first_pkt: &L4Pdu) -> Self {
       /* ... */
   }

    #[datatype_fn]
    fn update(&mut self, pdu: &L4Pdu) {
        /* ... */
    }
}
```

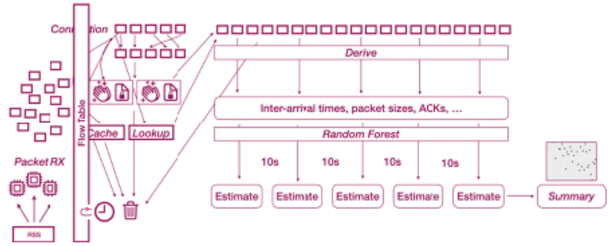
```rust
#[callback("tls and video_lookup")]
struct ResolutionEst {
    resolutions: Vec<usize>,
}

// impl ResolutionEst
#[callback_fn]
fn update(
    &mut self, feats: &FeatureChunk
) -> bool {
    /* ... */
}

#[callback_fn("ConnTerminated")]
fn end(&mut self) { /* ... */ }
```

User Rust Code



Iris Compiler

Merge, Deduplicate, Simplify



Iris Runtime

Many Concurrent Workloads with Minimal Aggregate Work

Iris

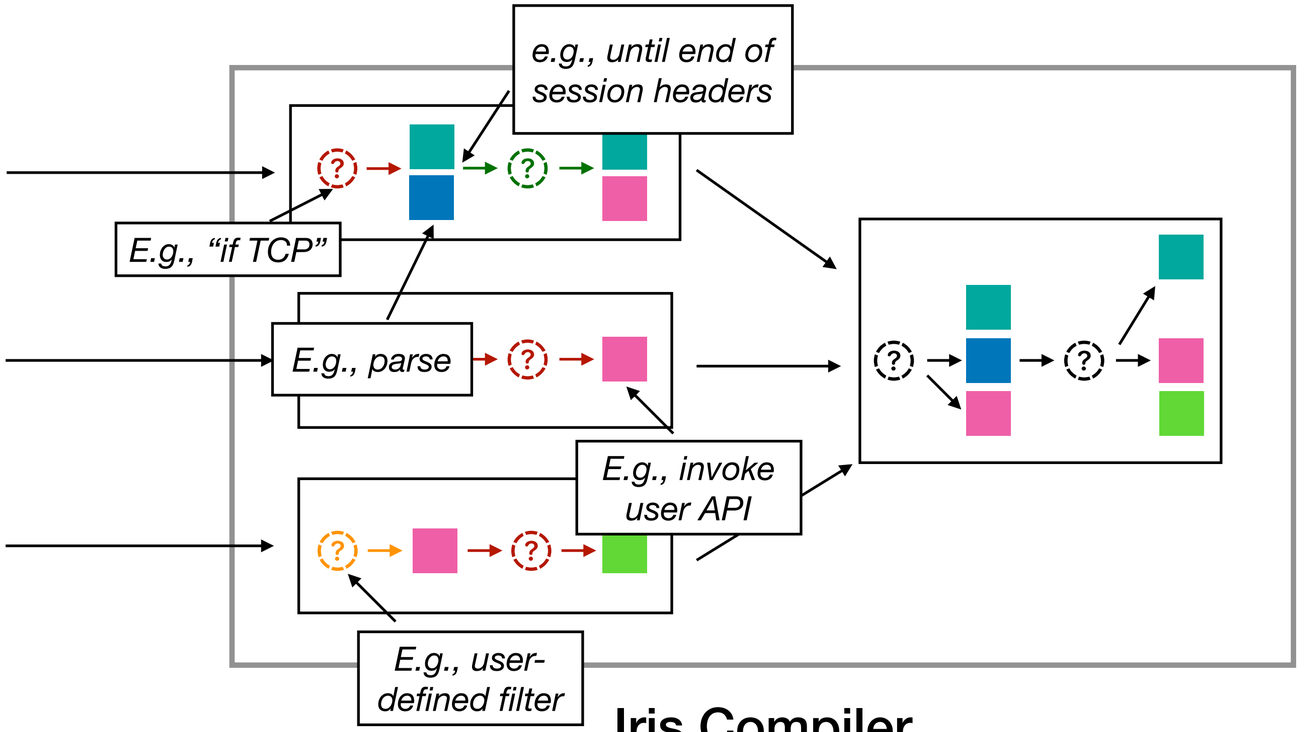
Compile-Time Optimization Enables High-Performance, Expressive Traffic Analysis

```rust
fn video_lookup(
    &mut self, tls: &TlsHandshake,
    ft: &FiveTuple
) -> FilterResult {
    /* ... */
}
```

```rust
#[datatype]
struct FeatureChunk { /* ... */ }

impl FeatureChunk {
   fn new(first_pkt: &L4Pdu) -> Self {
       /* ... */
   }

    #[datatype_fn]
    fn update(&mut self, pdu: &L4Pdu) {
        /* ... */
    }
}
```

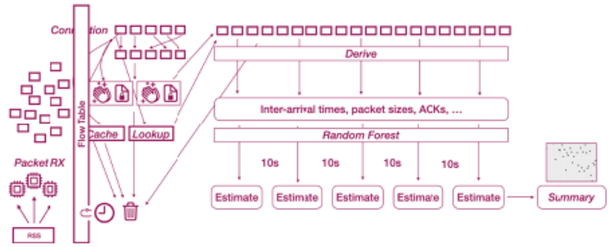
```rust
#[callback("tls and video_lookup")]
struct ResolutionEst {
    resolutions: Vec<usize>,
}

// impl ResolutionEst
#[callback_fn]
fn update(
    &mut self, feats: &FeatureChunk
) -> bool {
    /* ... */
}

#[callback_fn("ConnTerminated")]
fn end(&mut self) { /* ... */ }
```

User Rust Code



Iris Compiler

Merge, Deduplicate, Simplify



Iris Runtime

Many Concurrent Workloads with Minimal Aggregate Work

Iris Contributions

1. Flexibility

Developers write code in a general-purpose programming language

2. Abstractions

…over connection- and application-layer data

3. Performance

…while retaining enough structure for the compiler to optimize across analysis tasks.

Iris Contributions

1. Flexibility

Developers write code in a general-purpose programming language

2. Abstractions

…over connection- and application-layer data

3. Performance

…while retaining enough structure for the compiler to optimize across analysis tasks.

Programming Model

Compiler

Iris Contributions

1. Flexibility

Developers write code in a general-purpose programming language

2. Abstractions

…over connection- and application-layer data

Programming Model

3. Performance

…while retaining enough structure for the compiler to optimize across analysis tasks.

Compiler

Iris Directly Exposes Common Types

Subscription Interface:
Filter + Data Type(s) + Callback

```
#[callback("tls.sni contains 'nflx'")]
fn log_tls(tls: &TlsHandshake, ft: &FiveTuple) {
    log::info!("{}: {}", ft.src_subnet(24), tls);
}
```

Iris Directly Exposes Common Types

Subscription Interface:

Filter + Data Type(s) + Callback

```
#[callback("tls.sni contains 'nflx'")]
fn log_tls(tls: &TlsHandshake, ft: &FiveTuple) {
    log::info!("{}: {}", ft.src_subnet(24), tls);
}
```

Iris Directly Exposes Common Types

Subscription Interface:
Filter + Data Type(s) + Callback

```
#[callback("tls.sni contains 'nflx'")]
fn log_tls(tls: &TlsHandshake, ft: &FiveTuple) {
    log::info!("{}: {}", ft.src_subnet(24), tls);
}
```

Iris Directly Exposes Common Types

Subscription Interface:
Filter + Data Type(s) + Callback

```
#[callback("tls.sni contains 'nflx'")]
fn log_tls(tls: &TlsHandshake, ft: &FiveTuple) {
    log::info!("{}: {}", ft.src_subnet(24), tls);
}
```

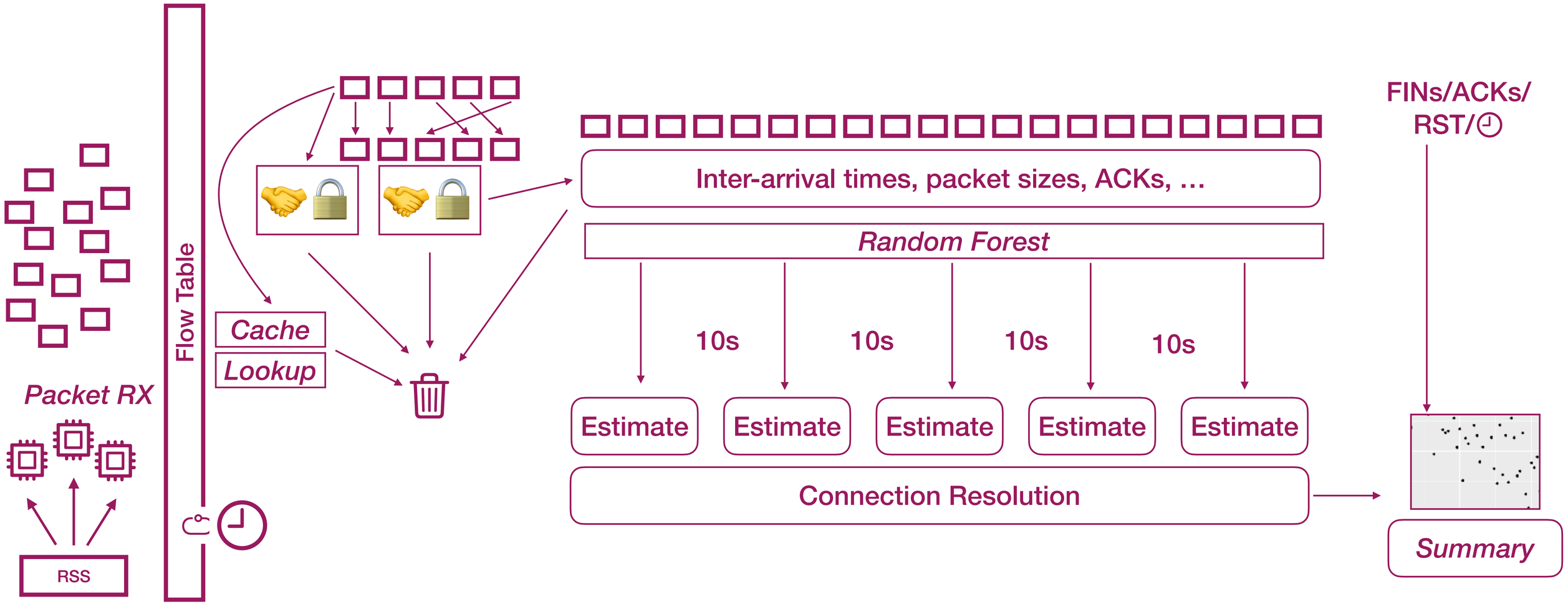
Iris Directly Exposes Common Types

Using Iris is Simple for Simple Use-Cases

```
#[callback("tls.sni contains 'nflx'")]
fn log_tls(tls: &TlsHandshake, ft: &FiveTuple) {
    log::info!("{}: {}", ft.src_subnet(24), tls);
}
```

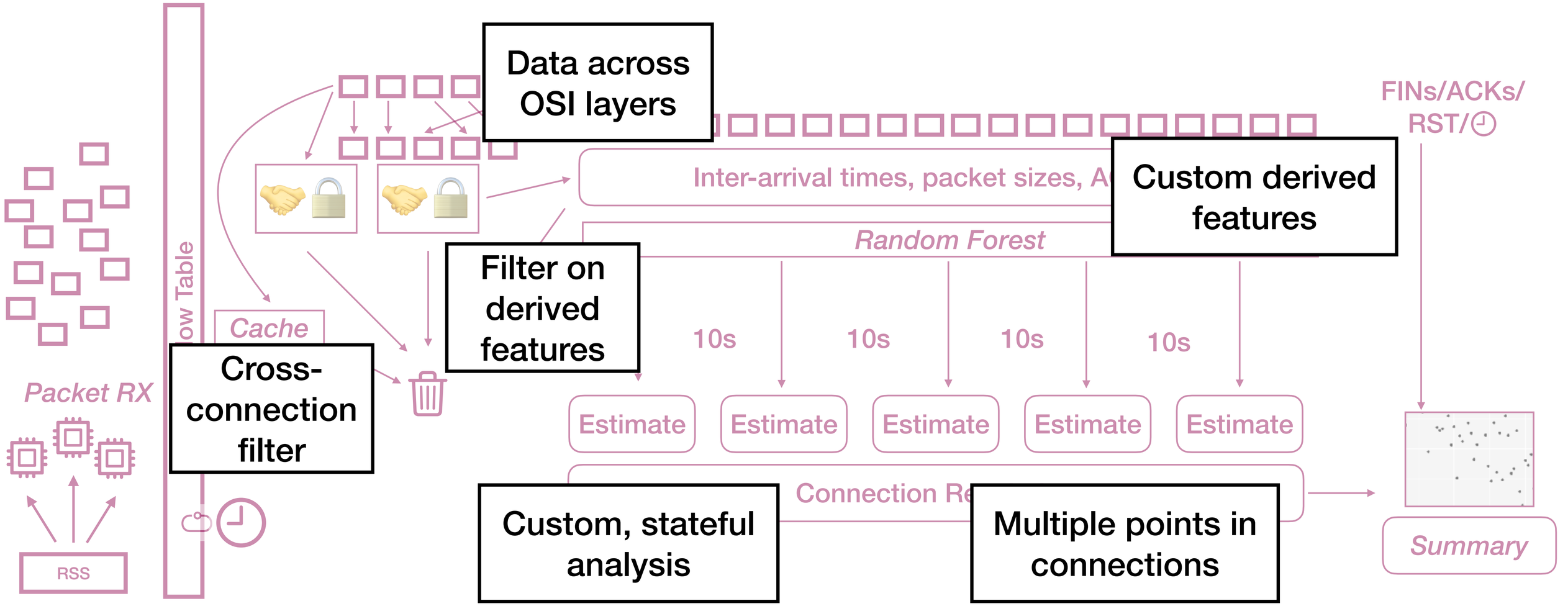
Iris Subscriptions are Extensible

Motivating Example: Video Resolution



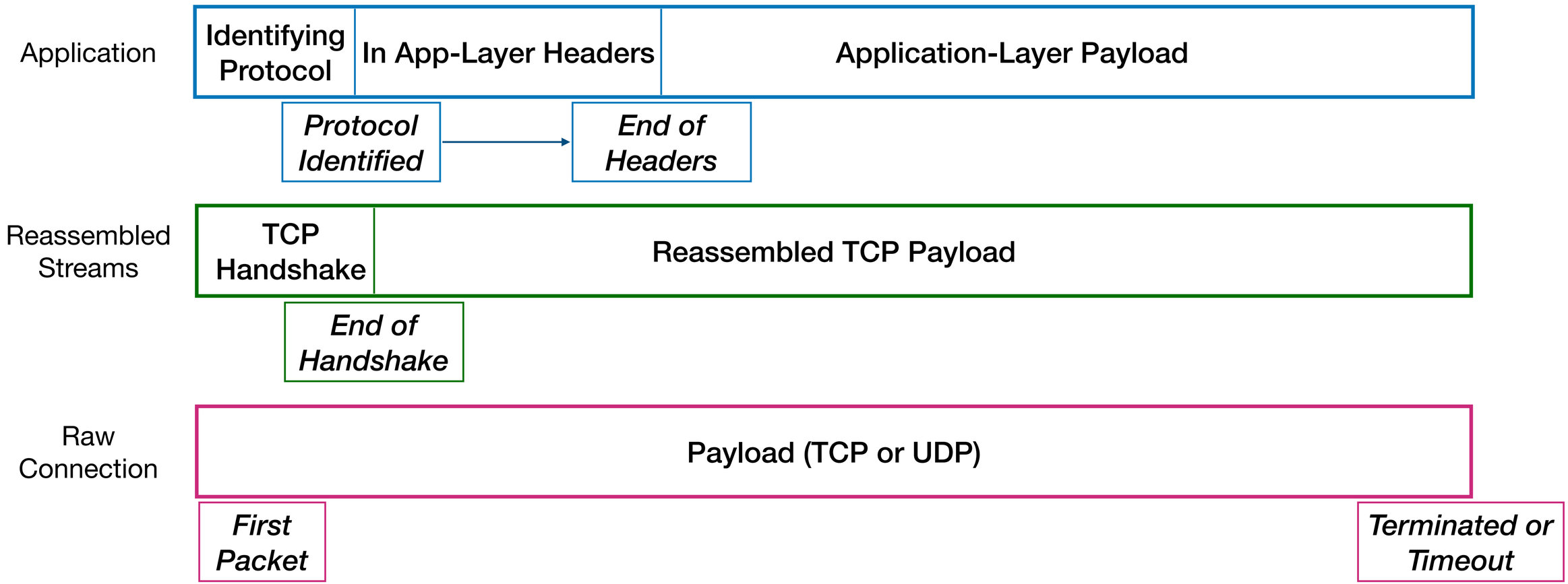
Iris Subscriptions are Extensible

Motivating Example: Video Resolution

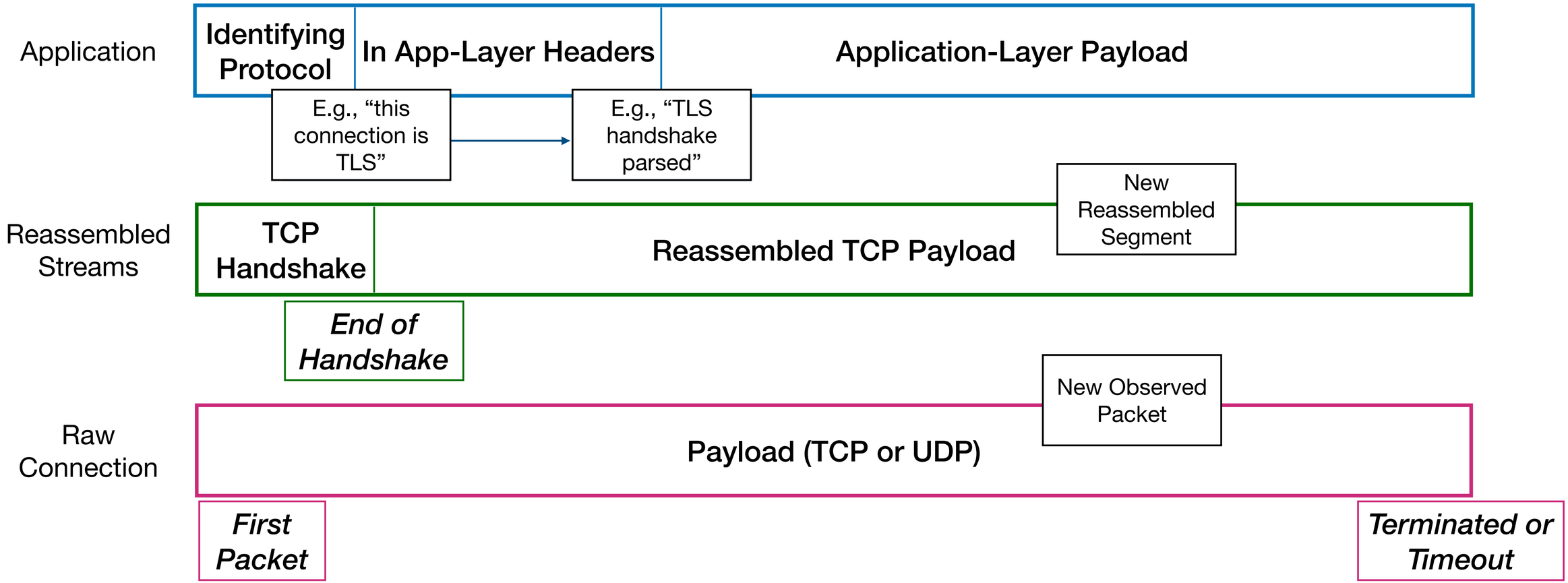


Iris allows users to define every subscription component in a general-purpose programming language.

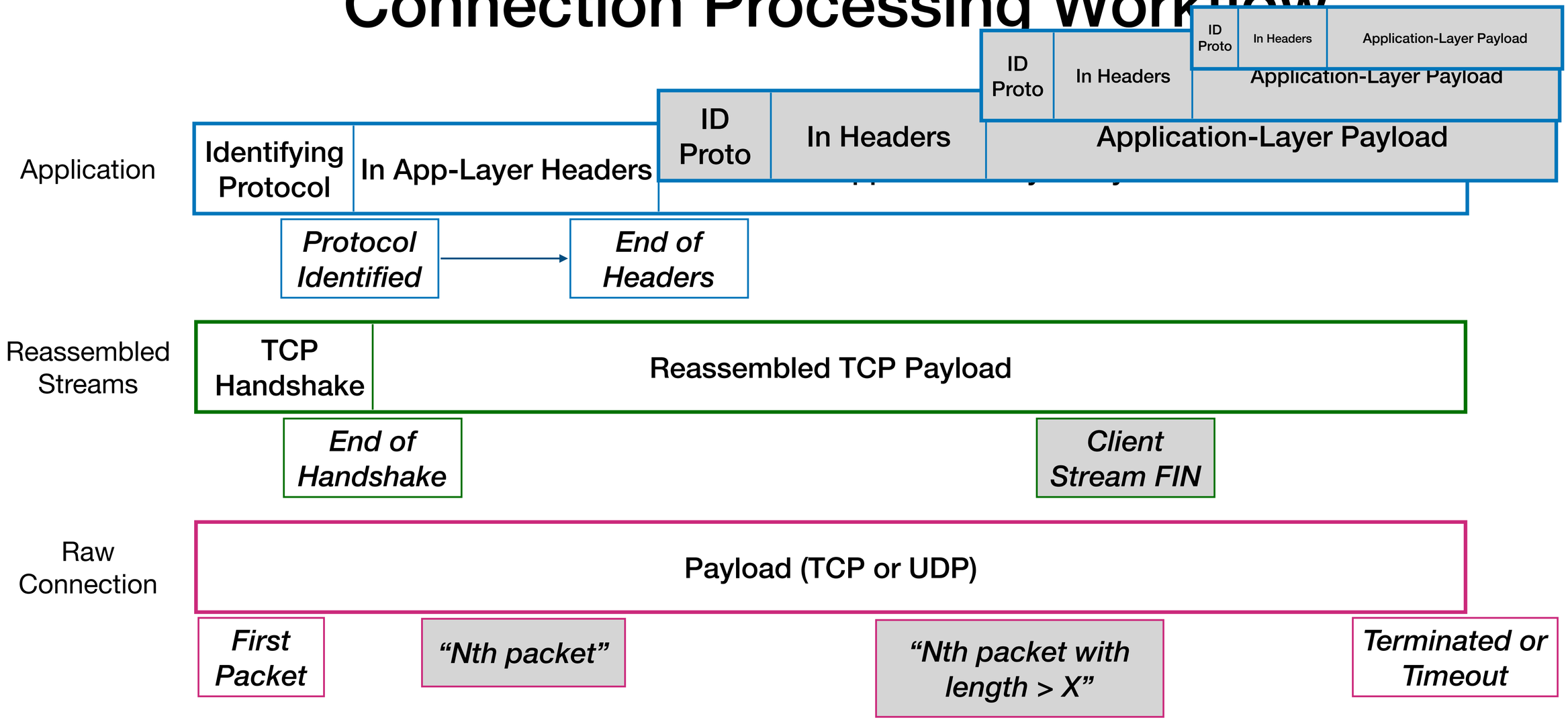
Users Define Subscriptions by Hooking into Iris' Connection Processing Workflow



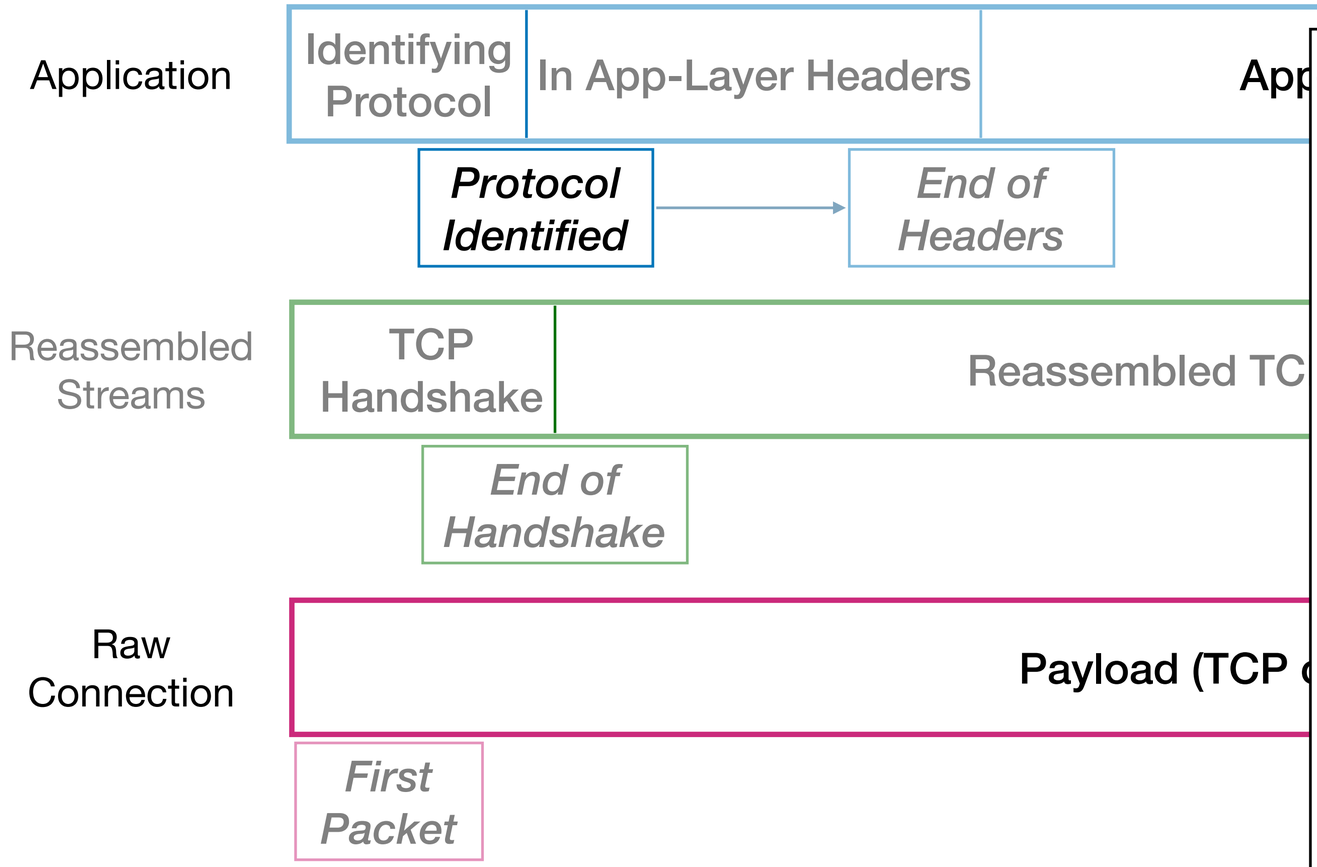
Users Define Subscriptions by Hooking into Iris' Connection Processing Workflow



Users Define Subscriptions by Hooking into Iris' Connection Processing Workflow

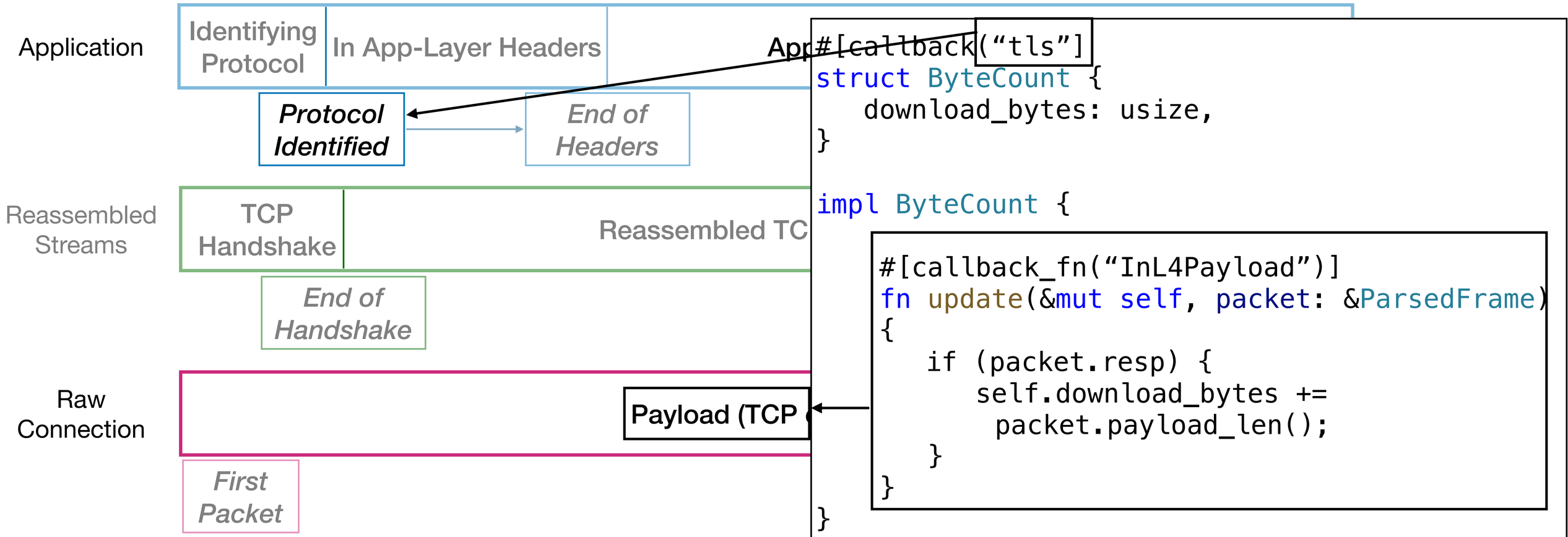


Users Define Subscriptions by Hooking into Iris' Connection Processing Workflow

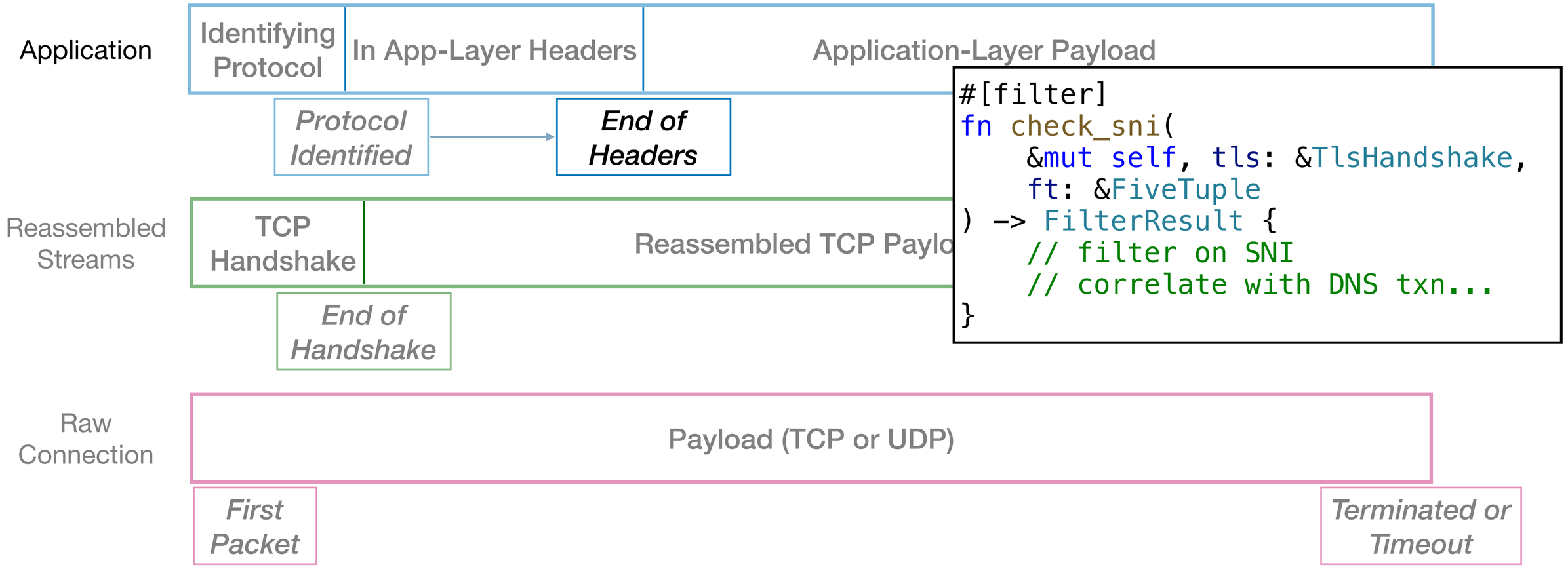



```
#[callback("tls")]
struct ByteCount {
   download_bytes: usize,
}

impl ByteCount {

    #[callback_fn("InL4Payload")]
    fn update(&mut self, packet: &ParsedFrame)
    {
        if (packet.resp) {
            self.download_bytes +=
              packet.payload_len();
        }
    }
}
```

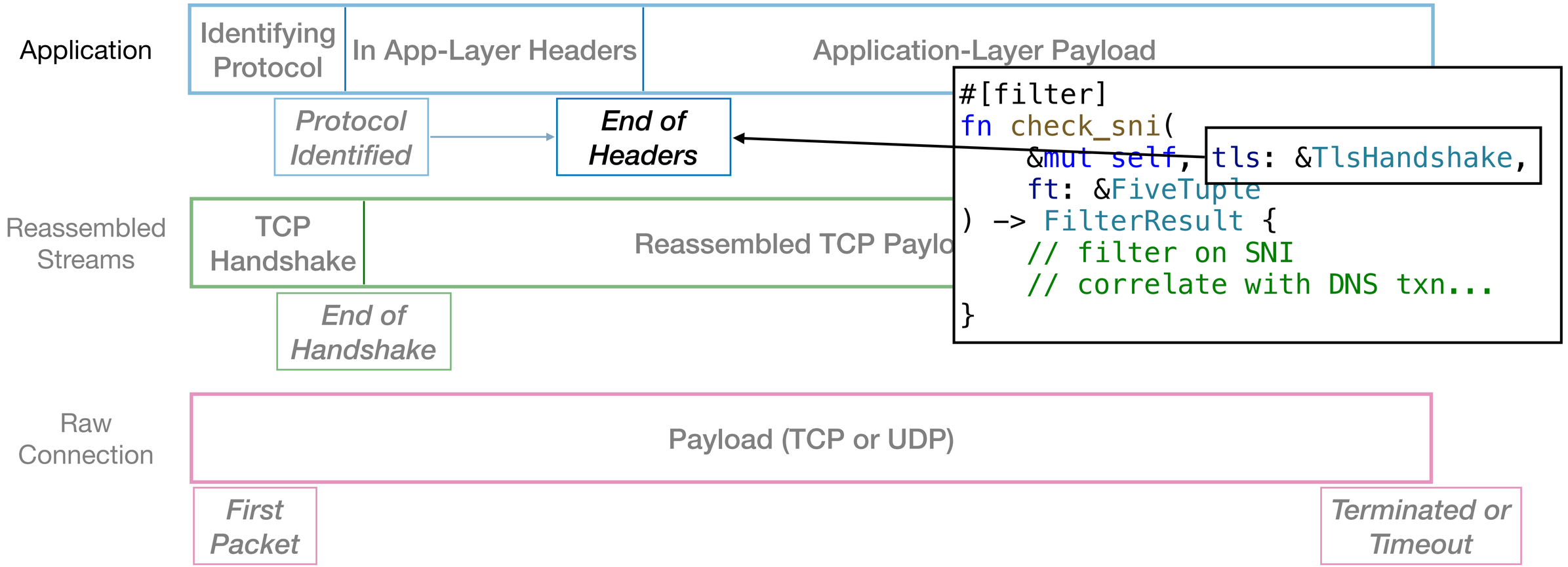
Users Define Subscriptions by Hooking into Iris' Connection Processing Workflow



Users Define Subscriptions by Hooking into Iris' Connection Processing Workflow



Users Define Subscriptions by Hooking into Iris' Connection Processing Workflow



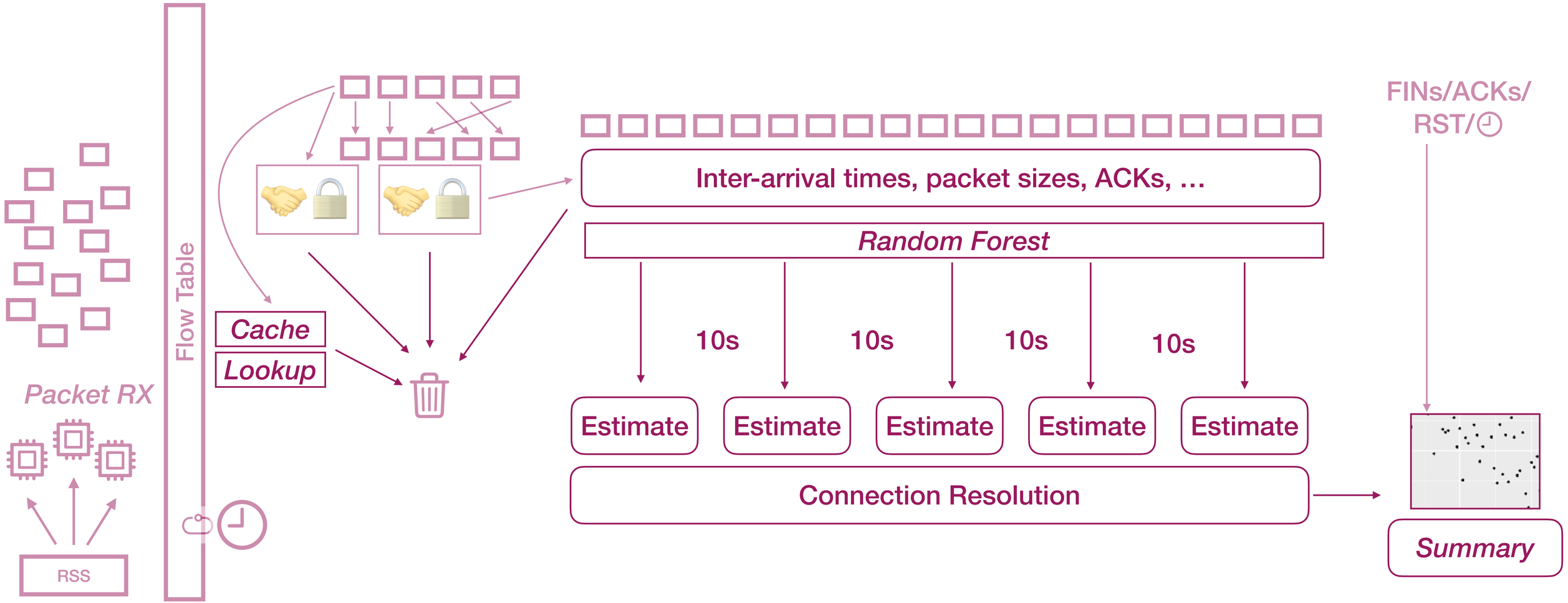
Users Compose Abstractions to Create More Complex Subscriptions

```
#[filter]
fn check_sni(
    &mut self, tls: &TlsHandshake,
    ft: &FiveTuple
) -> FilterResult {
    // filter on SNI
    // correlate with DNS txn...
}
```

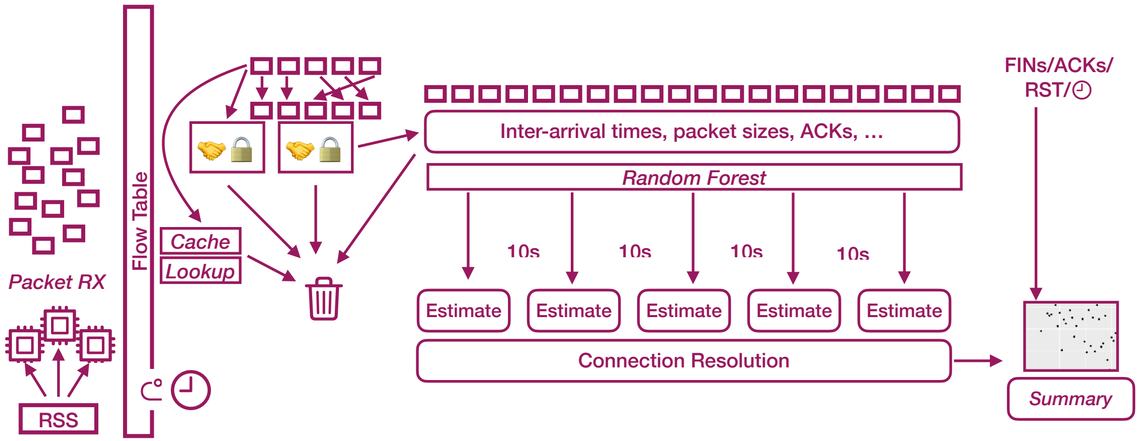
```
#[callback("tls and check_sni)"]
struct ByteCount {
   download_bytes: usize,
}

impl ByteCount {

    #[callback_fn("InL4Payload")]
    fn update(&mut self, packet: &ParsedFrame)
    {
        if (packet.resp) {
            self.download_bytes +=
             packet.payload_len();
        }
    }
}
```

Iris Implements and Composes Common Traffic Analysis Primitives

Motivating Example: Video Resolution



Before



Thousands of LoC

With Iris

~170 LoC

10 cores, >100Gbps

Motivating Example: Video Resolution

Iris Contributions

1. Flexibility

Developers write code in a general-purpose programming language

2. Abstractions

…over connection- and application-layer data

Programming Model

3. Performance

…while retaining enough structure for the compiler to optimize across analysis tasks.

Compiler

Iris Contributions

TL;DR: Construct and merge per-subscription control flow.

1. Flexibility

Developers write code in a general-purpose programming language

2. Abstractions

…over connection- and application-layer data

Programming Model

3. Performance

…while retaining enough structure for the compiler to optimize across analysis tasks.

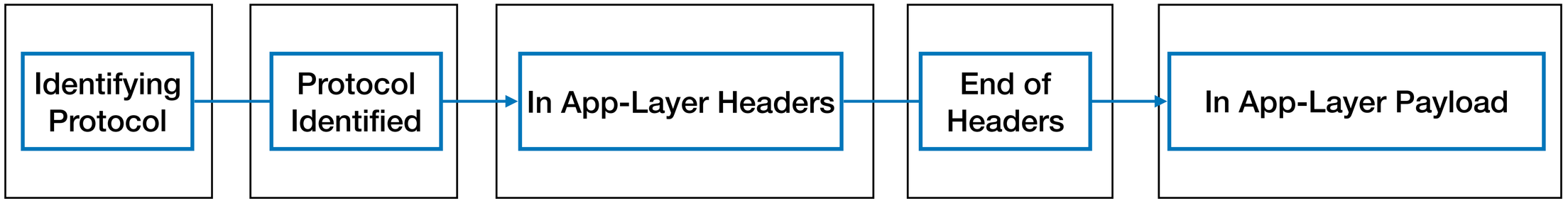
Compiler

By Mapping Subscriptions onto State Machines, Iris Builds Control Flow Graphs

⚠ Very simplified!
Additional layers omitted.

By Mapping Subscriptions onto State Machines, Iris Builds Control Flow Graphs

Treat state machines as incremental processing blocks



By Mapping Subscriptions onto State Machines, Iris Builds Control Flow Graphs

Subscription

```
#[callback("tls and video_lookup")]
struct ResolutionEst {
    resolutions: Vec<usize>,
}

// impl ResolutionEst
#[callback_fn]
fn update(
    &mut self, feats: &FeatureChunk
) -> bool {
    /* ... */
}

#[callback_fn("ConnTerminated")]
fn end(&mut self) { /* ... */ }
```

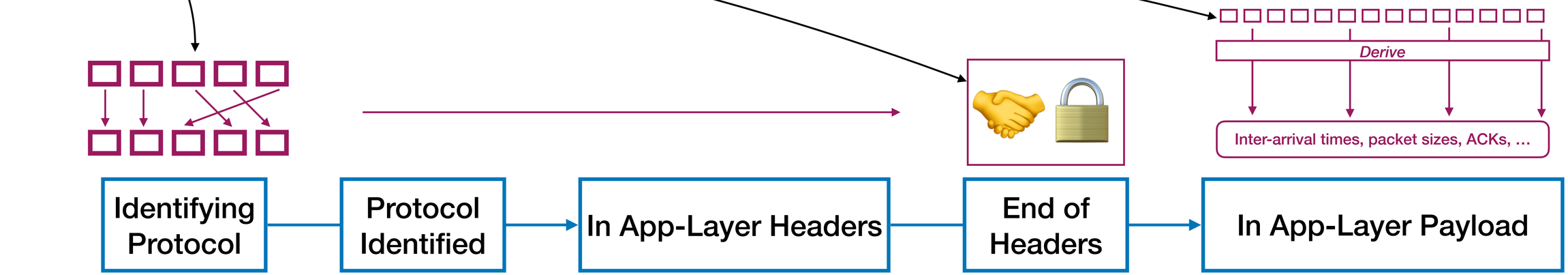
Map subscriptions to *actions* in state machines

Actions: incremental processing steps (at state/transitions)

By Mapping Subscriptions onto State Machines, Iris Builds Control Flow Graphs

Subscription

```rust
#[callback("tls and video_lookup")]
struct ResolutionEst {
    resolutions: Vec<usize>,
}

// impl ResolutionEst
#[callback_fn]
fn update(
    &mut self, feats: &FeatureChunk
) -> bool {
    /* ... */
}

#[callback_fn("ConnTerminated")]
fn end(&mut self) { /* ... */ }
```



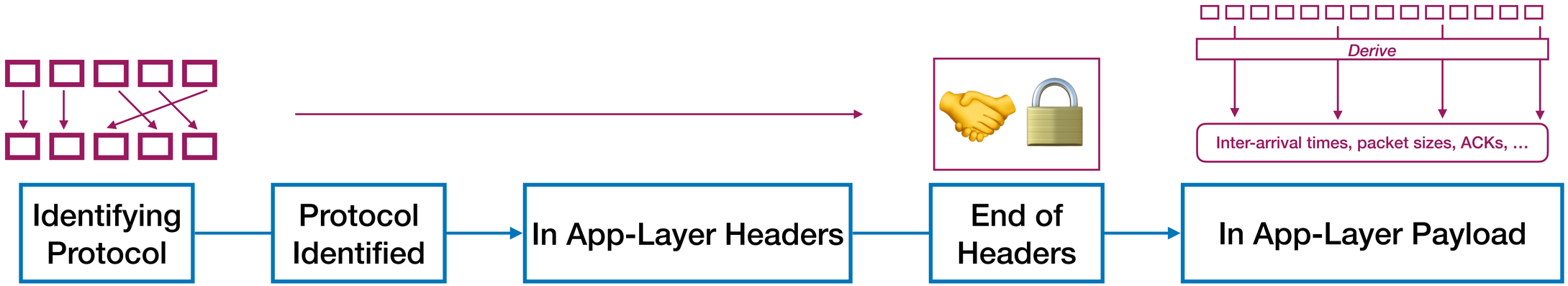
Map subscriptions to *actions* in state machines

Actions: incremental processing steps (at state/transitions)

By Mapping Subscriptions onto State Machines, Iris Builds Control Flow Graphs

Subscription

```
#[callback("tls and video_lookup")]
struct ResolutionEst {
    resolutions: Vec<usize>,
}

// impl ResolutionEst
#[callback_fn]
fn update(
    &mut self, feats: &FeatureChunk
) -> bool {
    /* ... */
}

#[callback_fn("ConnTerminated")]
fn end(&mut self) { /* ... */ }
```



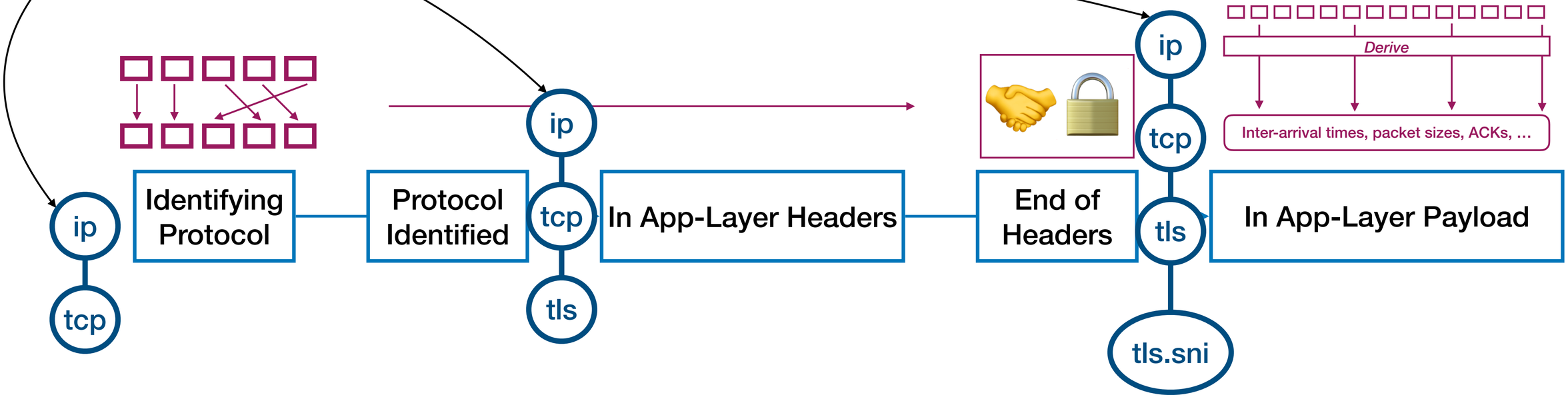
Incorporate subscription filters

Add incremental conditions to each processing step

By Mapping Subscriptions onto State Machines, Iris Builds Control Flow Graphs

Subscription

```
#[callback("tls and video_lookup")]
struct ResolutionEst {
    resolutions: Vec<usize>,
}

// impl ResolutionEst
#[callback_fn]
fn update(
    &mut self, feats: &FeatureChunk
) -> bool {
    /* ... */
}

#[callback_fn("ConnTerminated")]
fn end(&mut self) { /* ... */ }
```



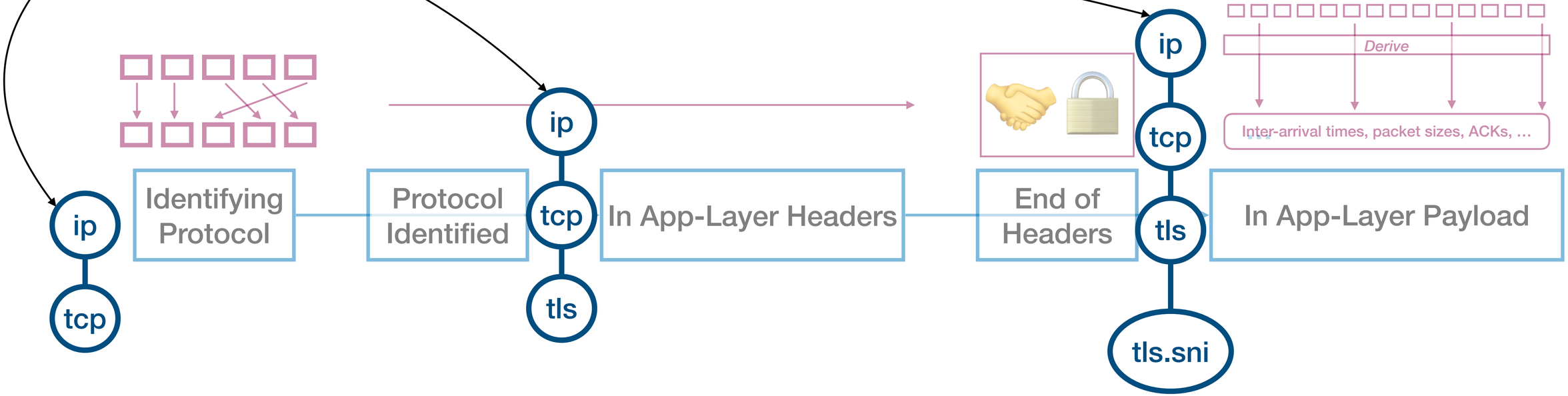
Incorporate subscription filters

Add incremental conditions to each processing step

By Mapping Subscriptions onto State Machines, Iris Builds Control Flow Graphs

Subscription

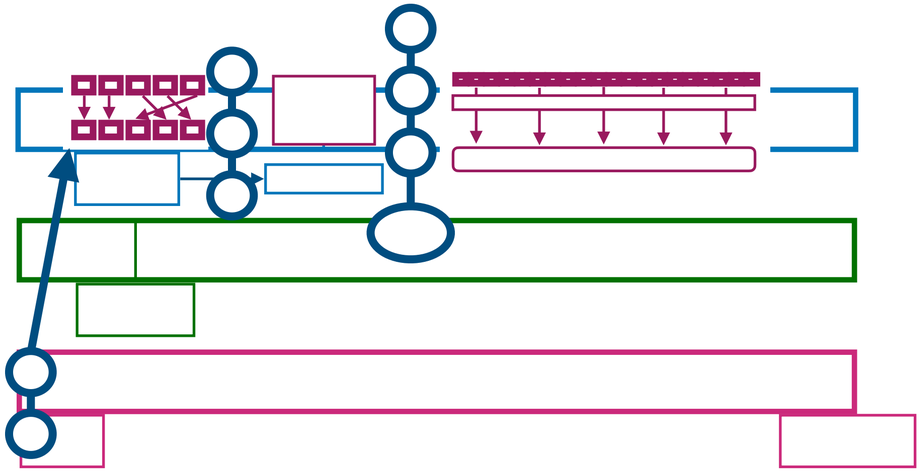
```
#[callback("tls and video_lookup")]
struct ResolutionEst {
    resolutions: Vec<usize>,
}

// impl ResolutionEst
#[callback_fn]
fn update(
    &mut self, feats: &FeatureChunk
) -> bool {
    /* ... */
}

#[callback_fn("ConnTerminated")]
fn end(&mut self) { /* ... */ }
```

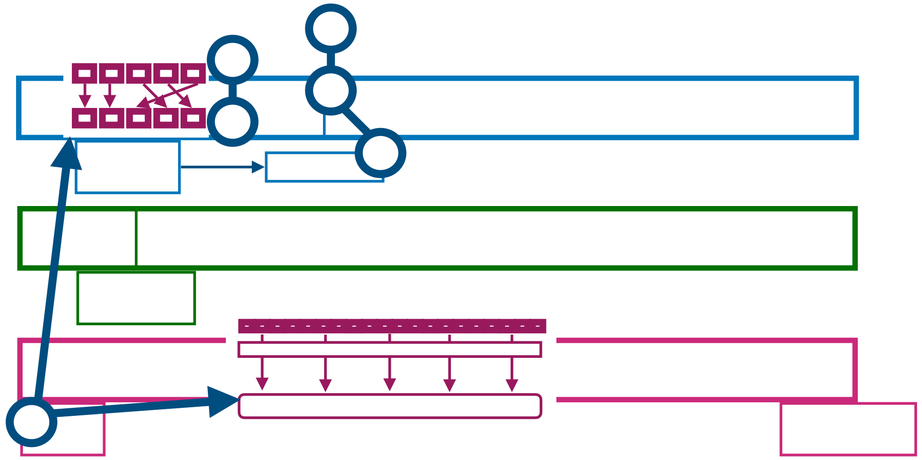


At runtime, Iris uses incremental filters to discard actions, data, and connections.

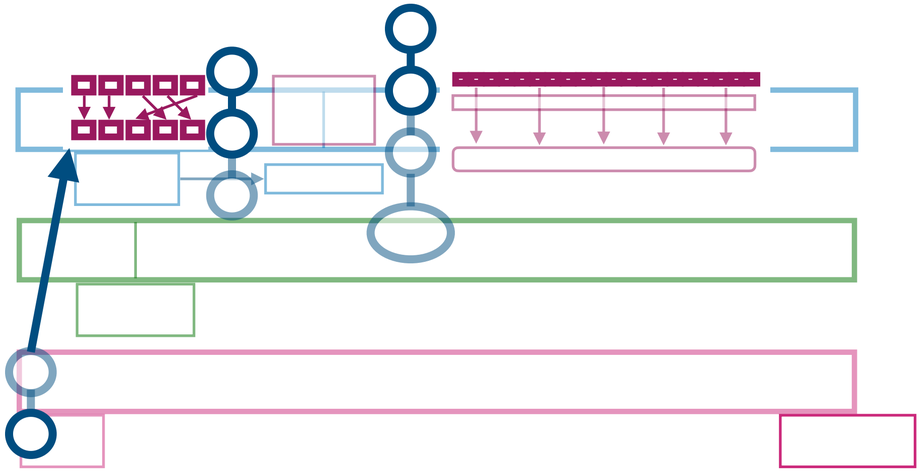
→ Minimal work necessary to fulfill subscriptions.

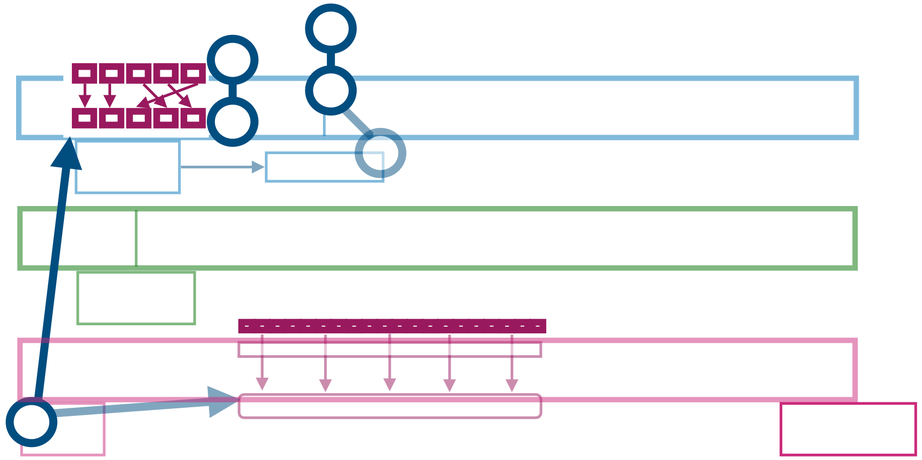
By Building Control Flow Graphs, Iris Can Merge Multiple Subscriptions





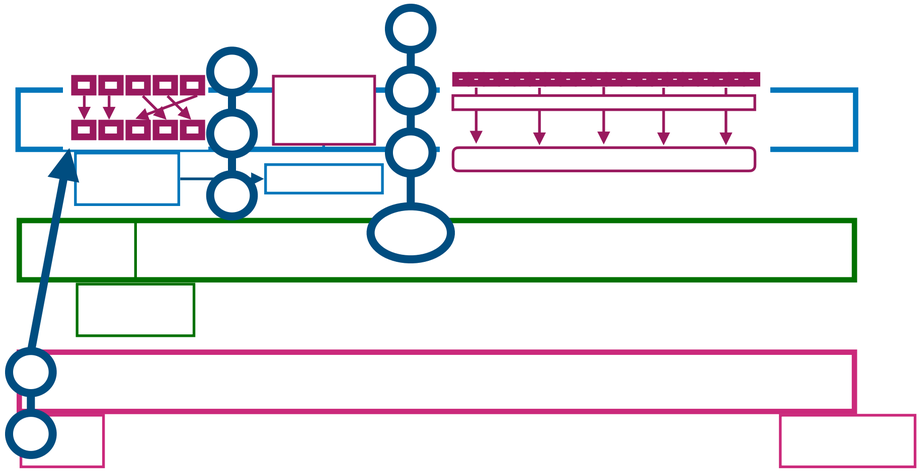
By Building Control Flow Graphs, Iris Can Merge Multiple Subscriptions

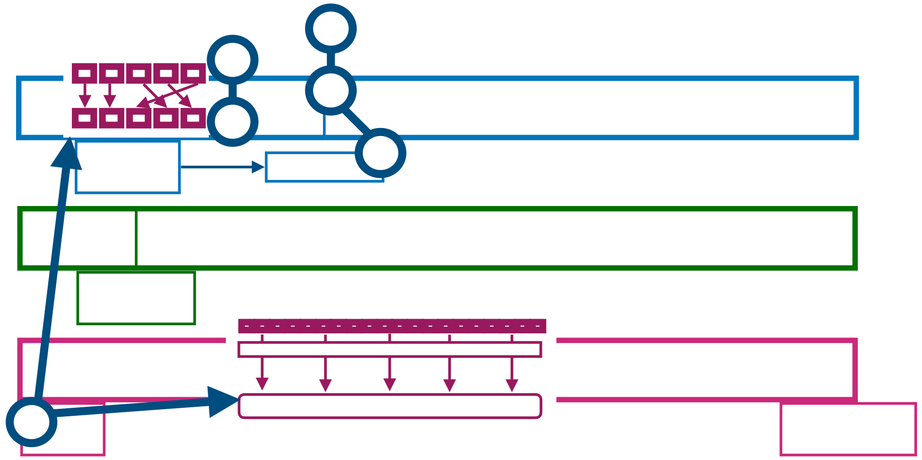




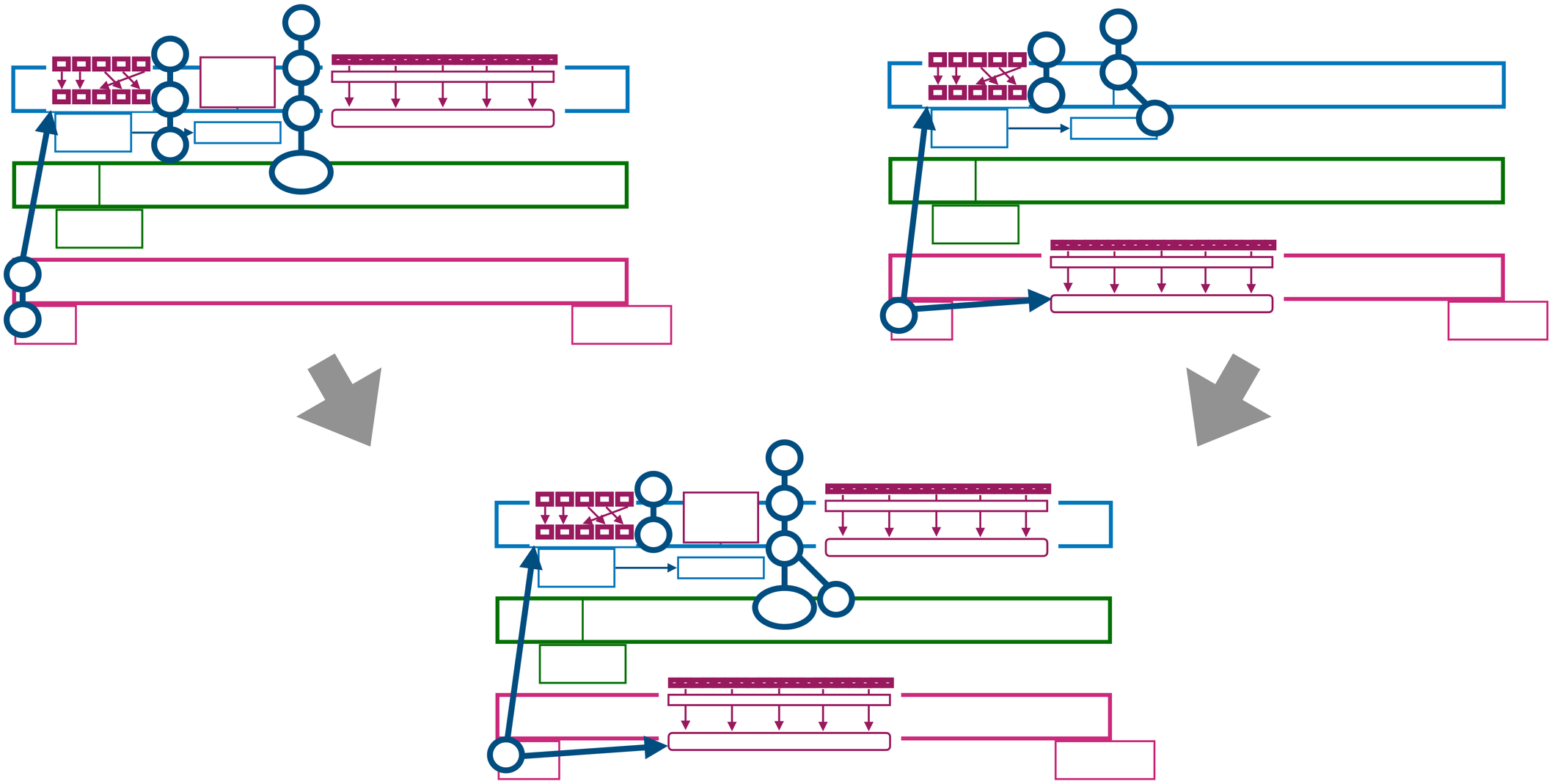
Subscriptions typically share some requirements.

By Building Control Flow Graphs, Iris Can Merge Multiple Subscriptions

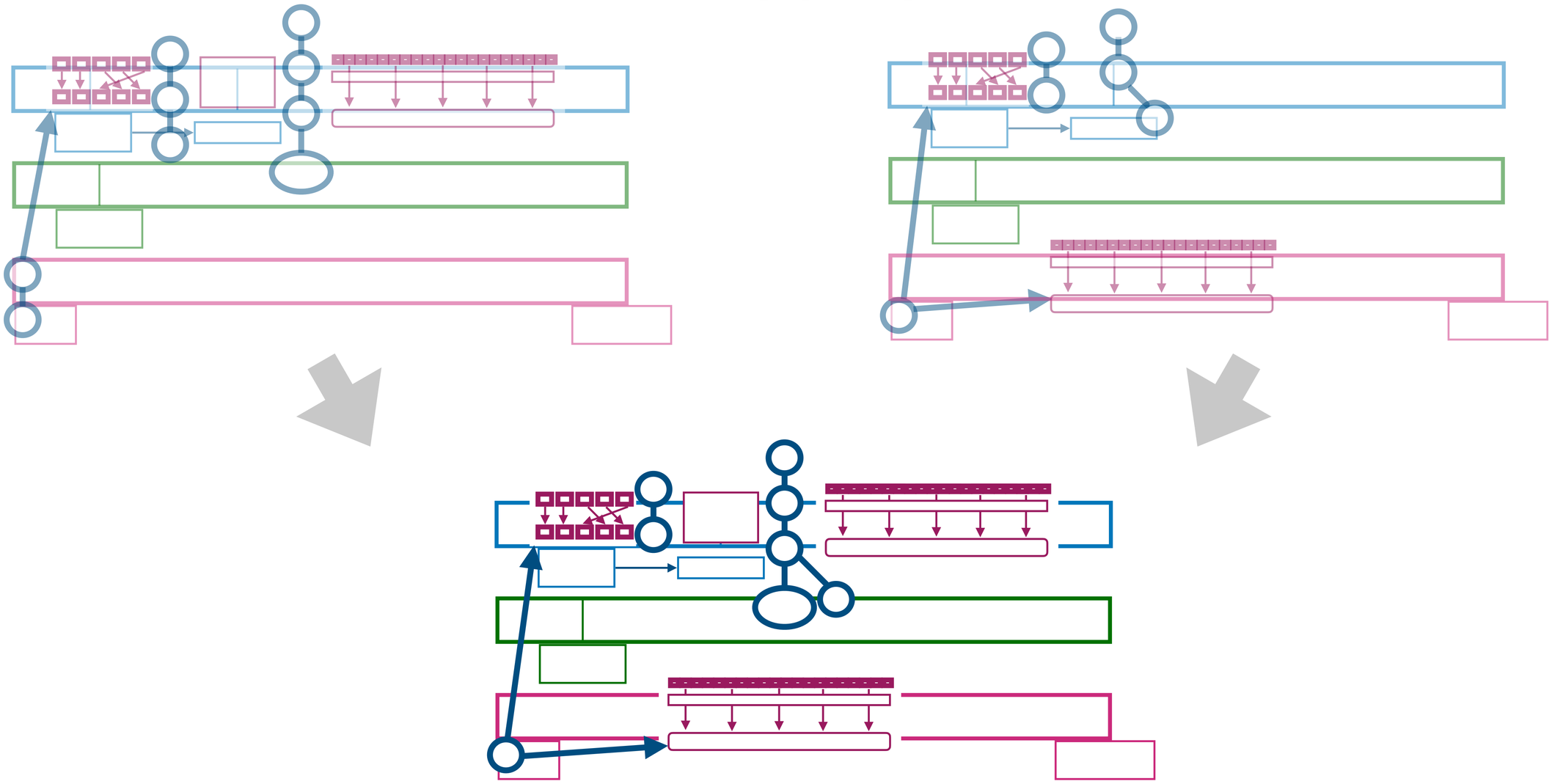




By Building Control Flow Graphs, Iris Can Merge Multiple Subscriptions



Multiple Subscriptions
Minimal Aggregate Work



Iris Evaluation

Iris Compiler: Performance

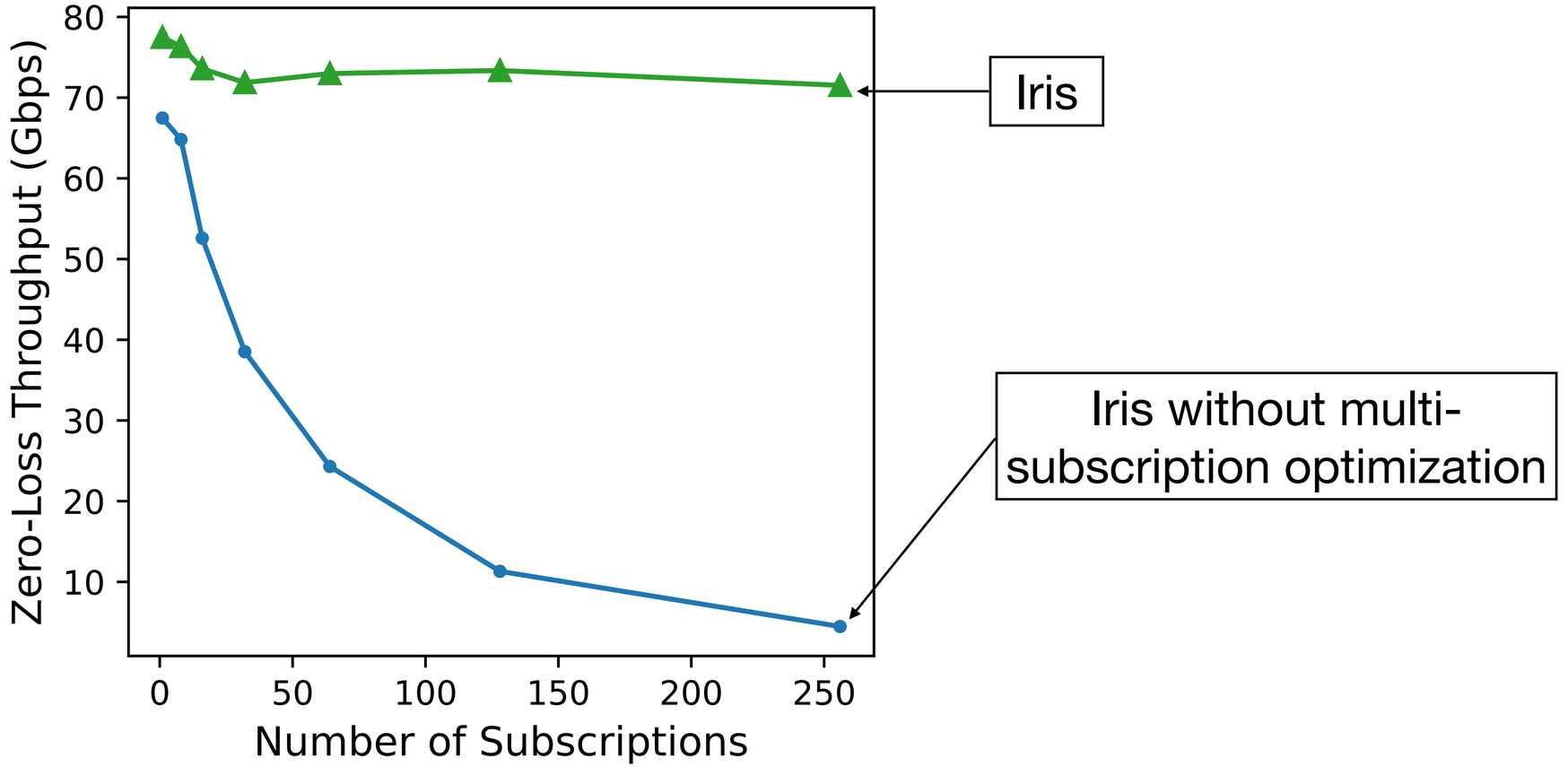
- Does Iris scale to many subscriptions (analysis tasks)?
- Does Iris effectively share work across subscriptions?

Interface: Flexibility + High-Level Abstractions

- Can we easily express complex, real-world use-cases in Iris?

Iris Scales to Hundreds of Subscriptions

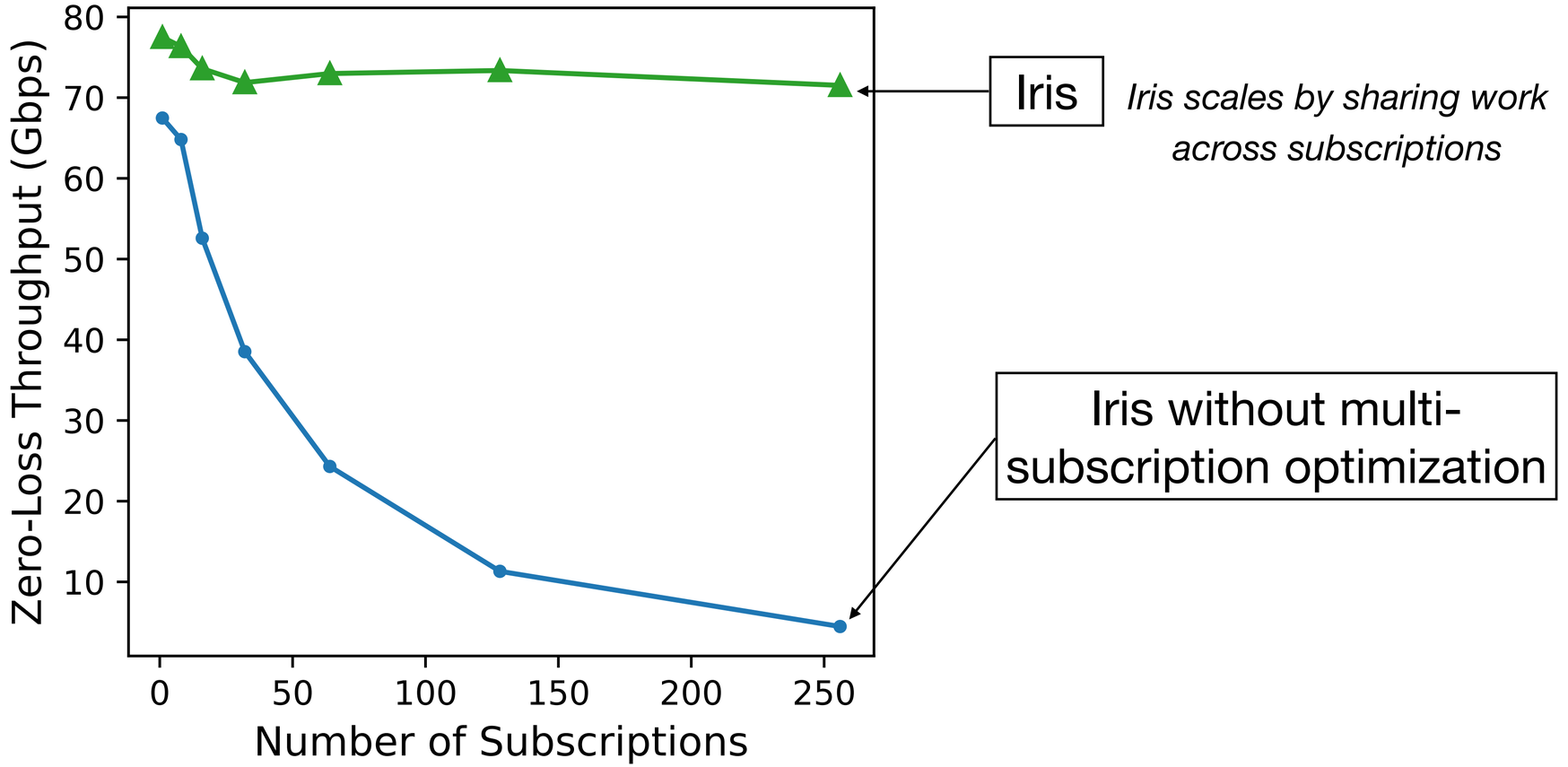
Evaluation on Stanford's network, two cores to force CPU bottleneck



HTTP transactions with *N* malicious hosts

Iris Scales to Hundreds of Subscriptions

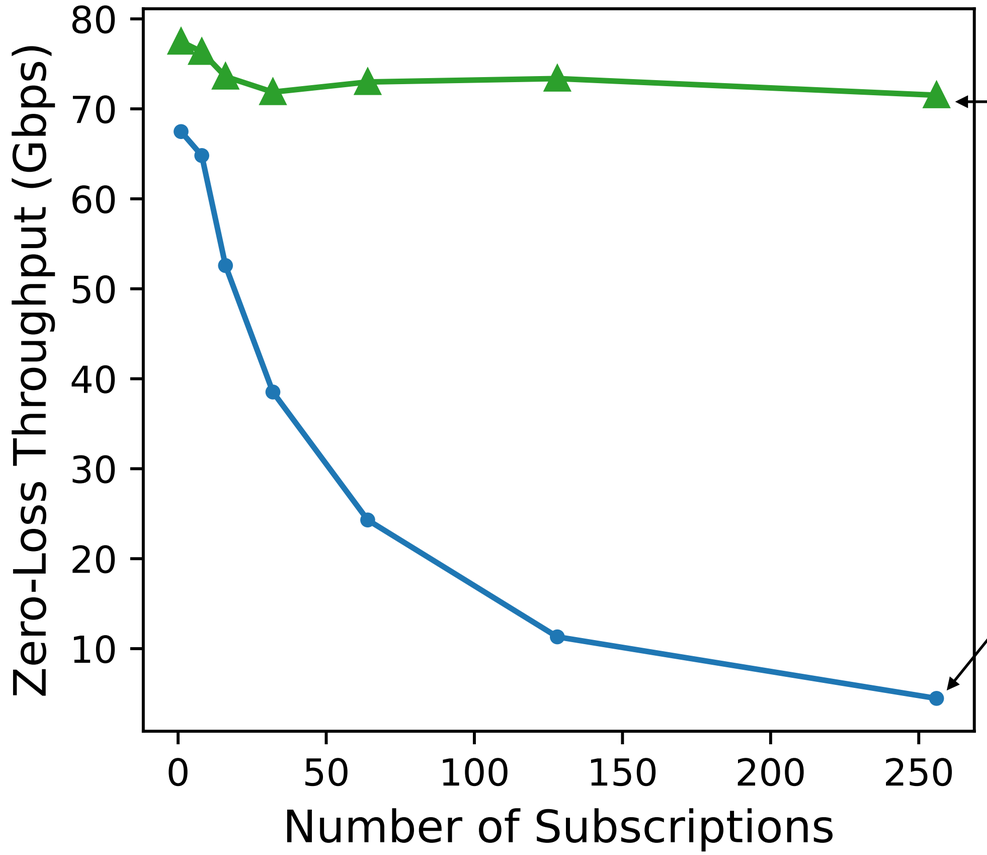
Evaluation on Stanford's network, two cores to force CPU bottleneck



HTTP transactions with *N* malicious hosts

Iris Scales to Hundreds of Subscriptions

Evaluation on Stanford's network, two cores to force CPU bottleneck



Iris

Iris scales by sharing work across subscriptions

Iris without multi-subscription optimization

Scaling multiple subscriptions <u>requires</u> sharing incremental work.

HTTP transactions with *N* malicious hosts

Evaluating Iris with Real-World Analysis

Measuring Security Practices *(DeKoven ’19)*

OpenVPN Censorship Risks *(Xue ’22)*

Estimating Video Resolution *(Bronzino ’19)*

All concurrently

Evaluating Iris with Real-World Analysis

Measuring Security Practices *(DeKoven ’19)*

Application-layer parsing

OpenVPN Censorship Risks *(Xue ’22)*

Custom data type

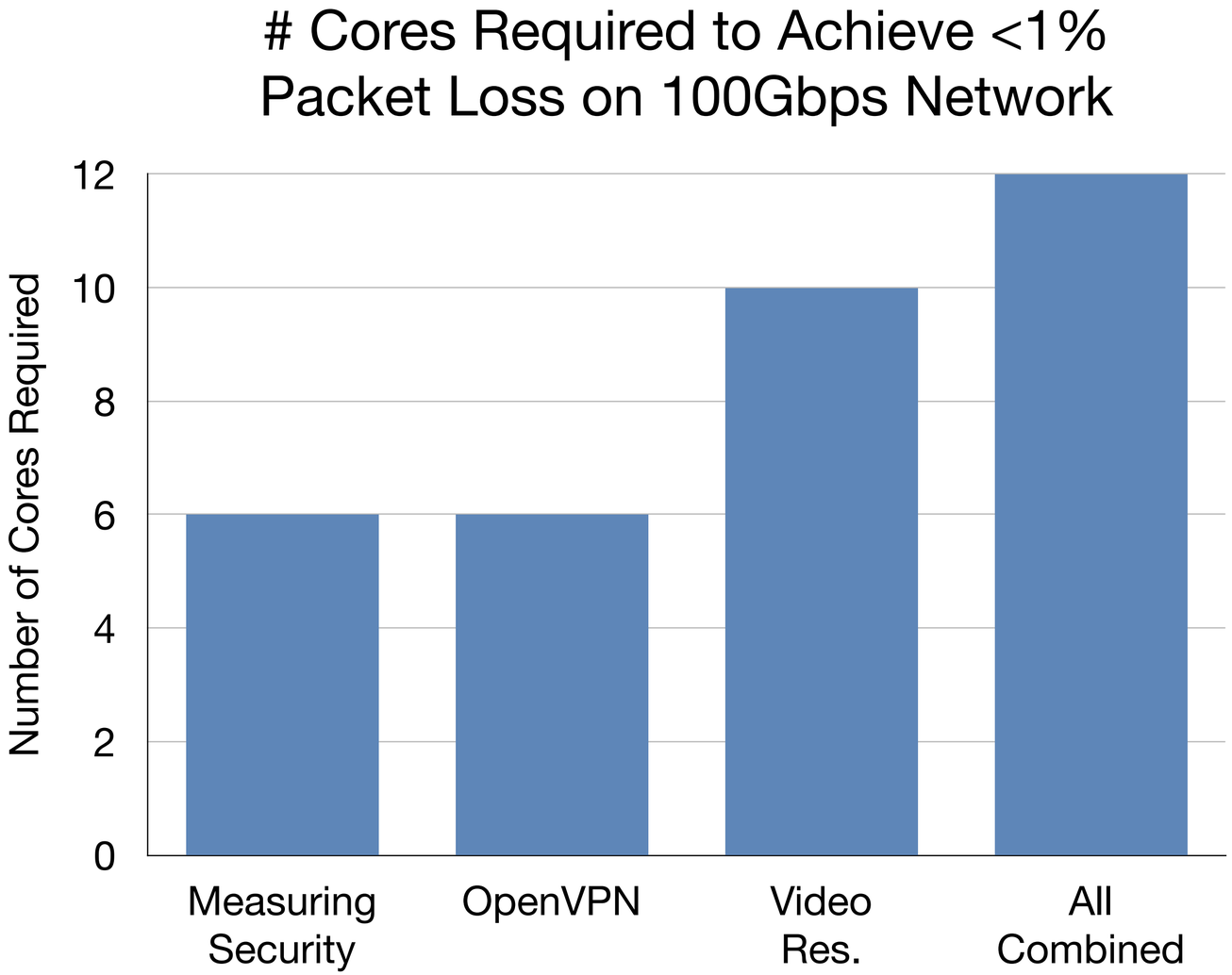
Estimating Video Resolution *(Bronzino ’19)*

Complex callback

All concurrently

Multiple heterogeneous subscriptions

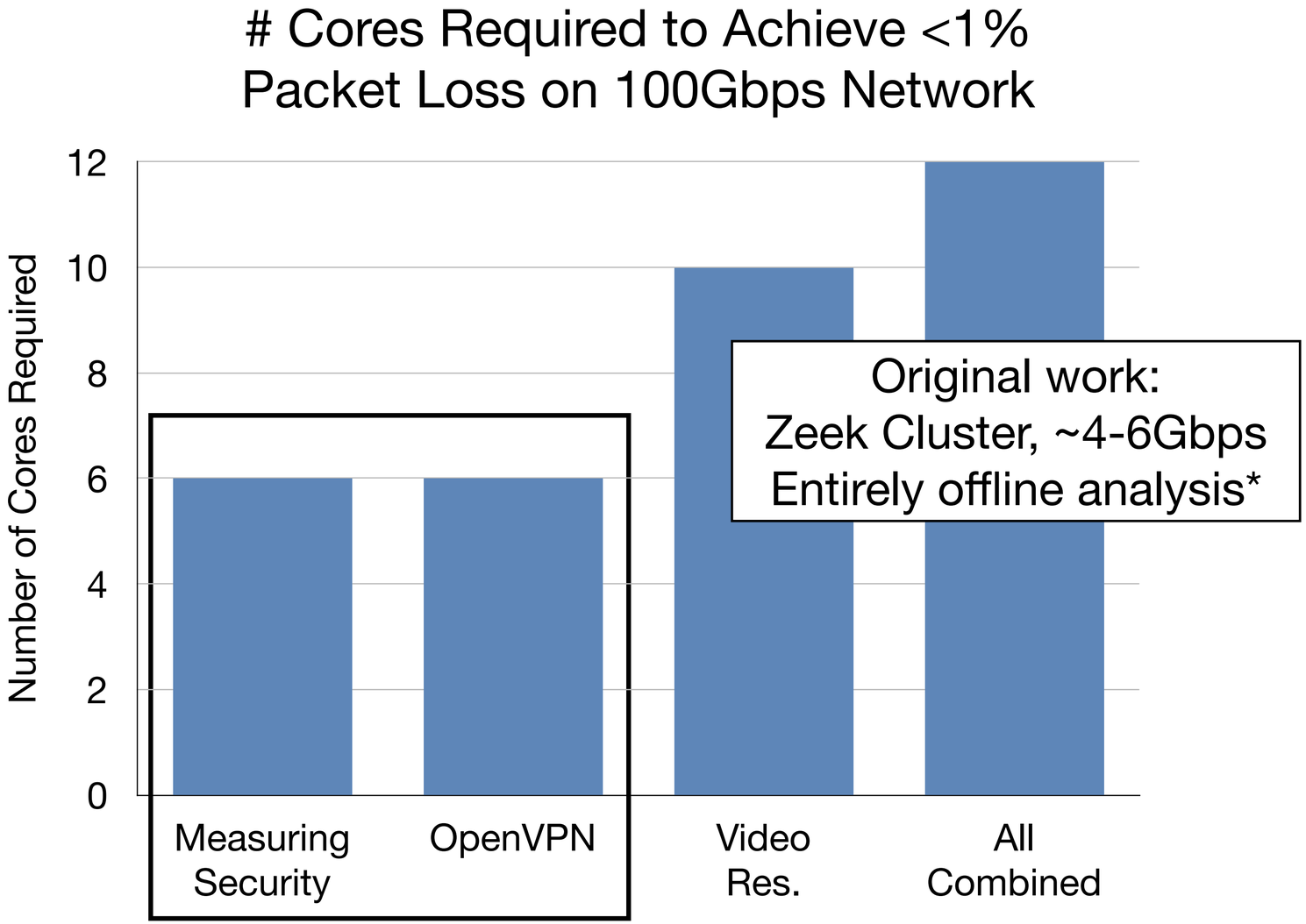
Iris Runs Complex Experiments at Line Rate



Lines of Code Required to Implement in Iris

Measuring Security	~200 LoC
OpenVPN	~230 LoC
Video Resolution	~170 LoC

Iris Runs Complex Experiments at Line Rate

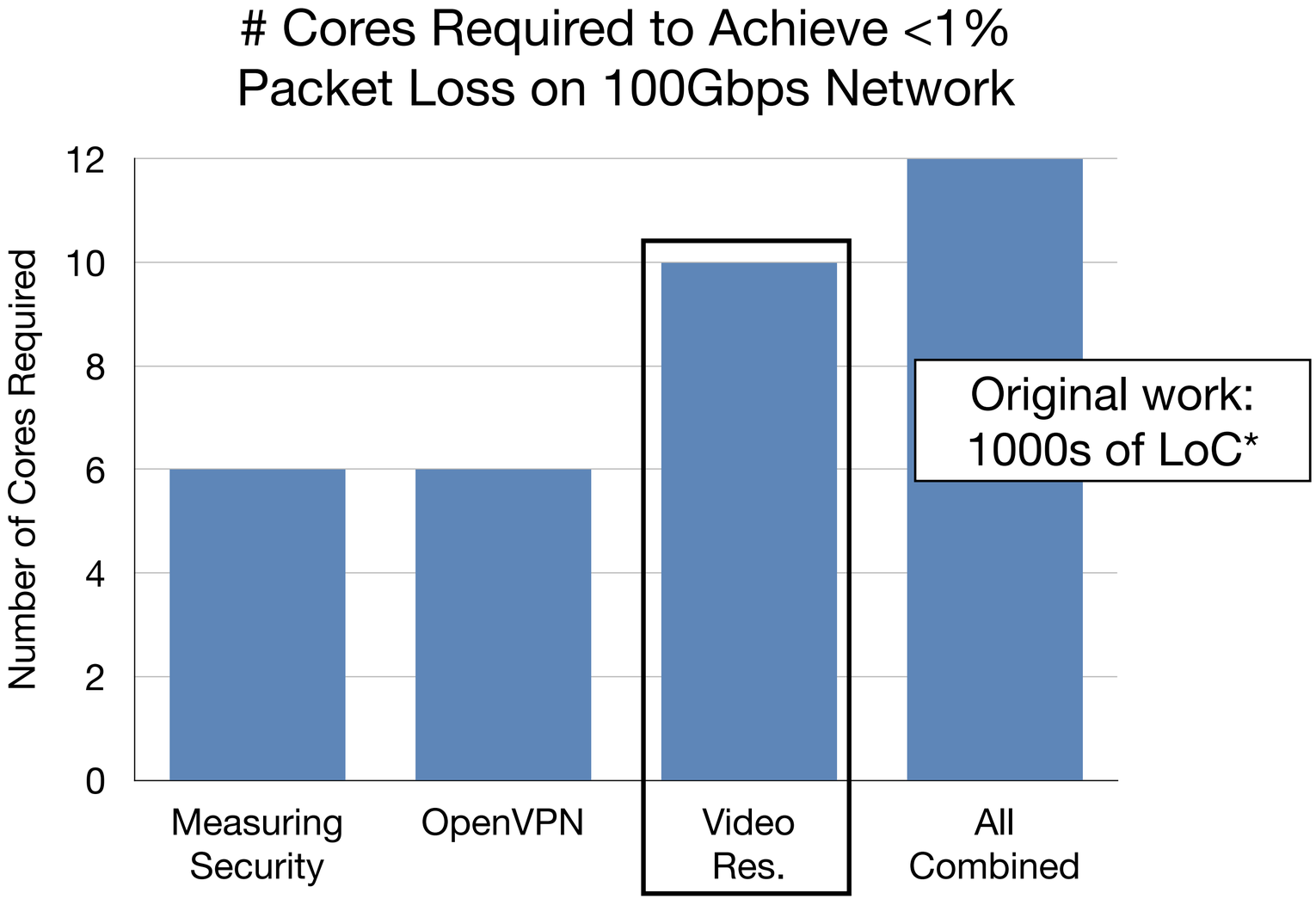


Lines of Code Required to Implement in Iris

Measuring Security	~200 LoC
OpenVPN	~230 LoC
Video Resolution	~170 LoC

*We were unable to replicate these studies in Zeek given our storage and compute constraints

Iris Reduces Developer Effort

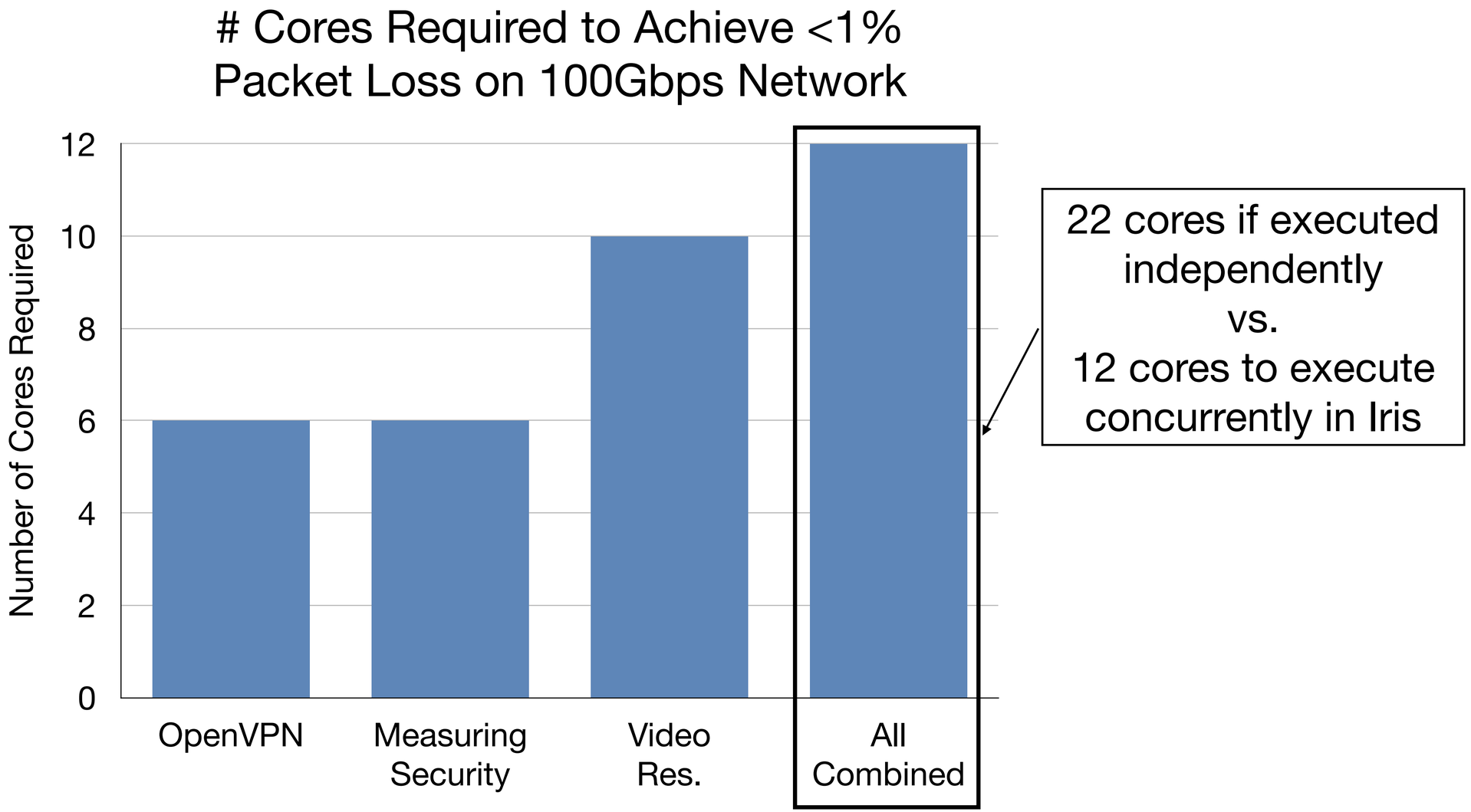


Lines of Code Required to Implement in Iris

Measuring Security	~200 LoC
OpenVPN	~230 LoC
Video Resolution	~170 LoC

*Original artifact not available; preliminary version of Traffic Refinery, which has ~6K LoC

Iris Optimizes Across Multiple Subscriptions



Iris: Expressive Traffic Analysis for the Modern Internet

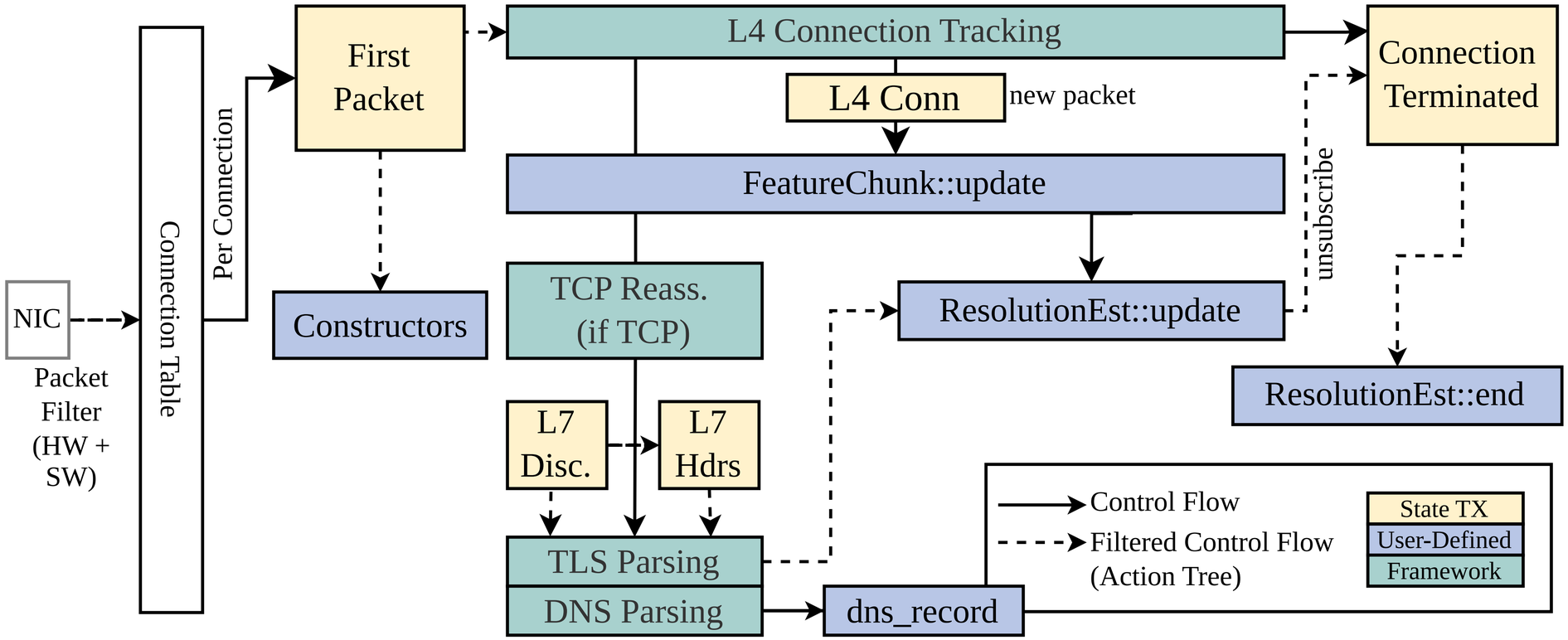
Thea Rossman, Diana Qing, Gerry Wan, Zakir Durumeric
Stanford University

https://github.com/stanford-esrg/iris

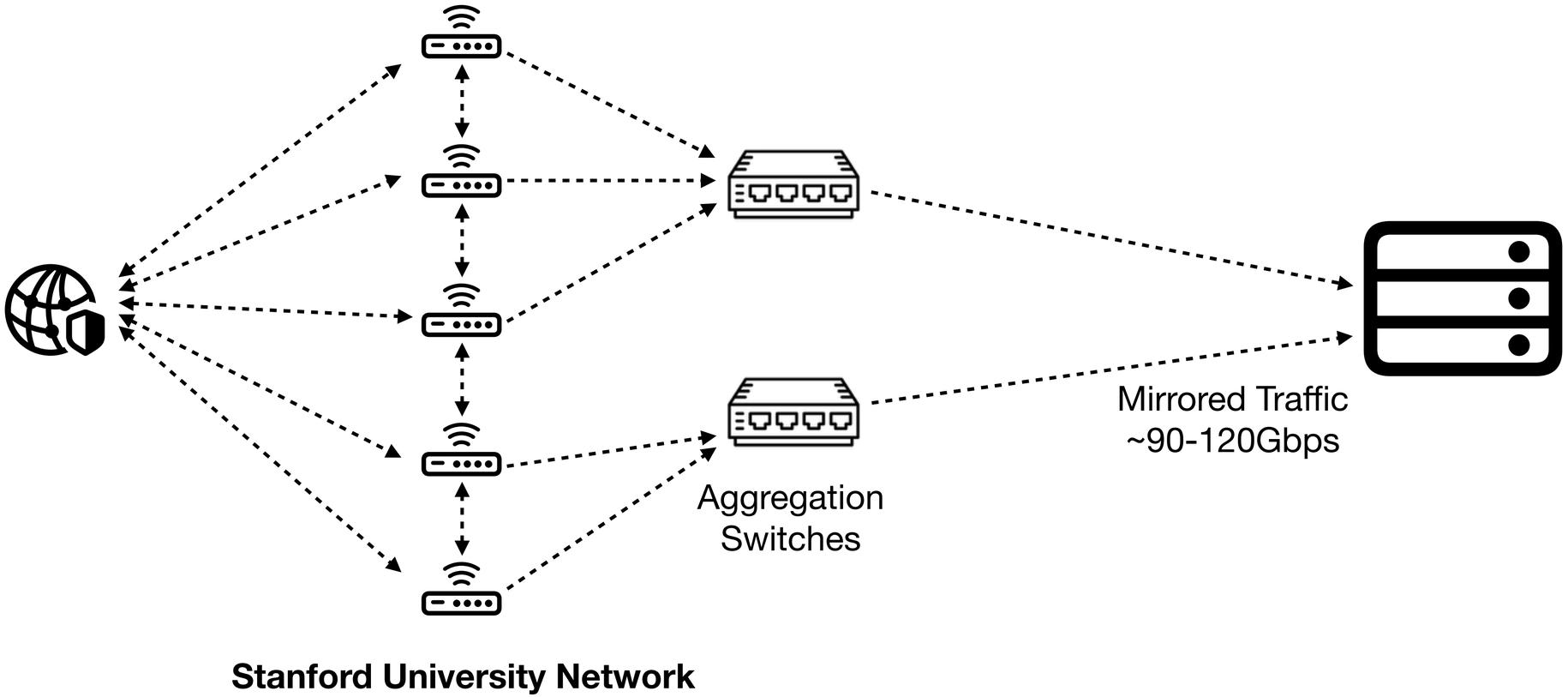
Contact: thea.rossman@cs.stanford.edu

Additional Slides

Generated Runtime Pipeline



Experiment Setup



Real-World Case Studies

	Max. Memory Utilization (App)	Max. Memory Utilization (DPDK)
Measuring Security	3.9GB	2.2GB
OpenVPN	2.5GB	1.0GB
ML QoS	3.5GB	1.5GB
All	5.0GB	3.5GB

Compilation Time

- Scales with number and complexity of subscriptions
- Majority of time is Rust release optimizations
- Development (debug builds) are typically fast

	Debug	Release	Iris
Use Cases (all)	2.5s	30s	50ms
HTTP - One Subscr.	5s	30s	15ms
HTTP - 5k Subscr.	30s	120s	18s

Ethics

- **Ethics Review**: Stanford privacy office, network IT, IRB (out of scope)
- **Privacy**: anonymize IP addresses, minimize data collection, perform flow-level analysis online (when possible), delete data within 2 weeks, no DHCP
- **Impact on Users:** mirrored traffic only
- **Operational Security**: rigorous access controls on network tap
- **Dual Use**: not optimized for latency or dragnet monitoring, not introducing fundamentally new capabilities