It's 11PM.

DO YOU KNOW WHERE YOUR HEADERS ARE?

Measuring Network Privacy with

Share Count Analysis

David Naylor

Peter Steenkiste

Carnegie Mellon University

GOAL

measure “how private” a network architecture or protocol is

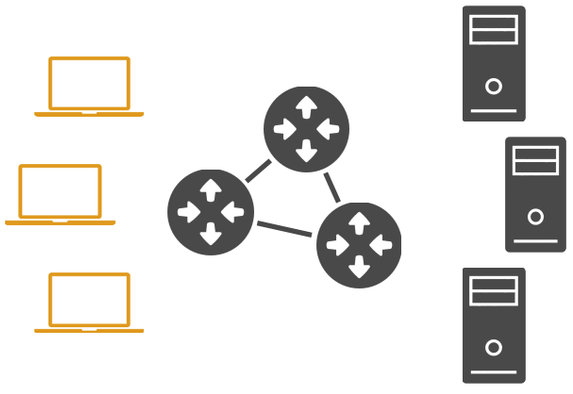
GOAL

measure "how private" a network architecture or protocol is

1 What is Privacy? | 2 Threat Model | 3 "Indicators"

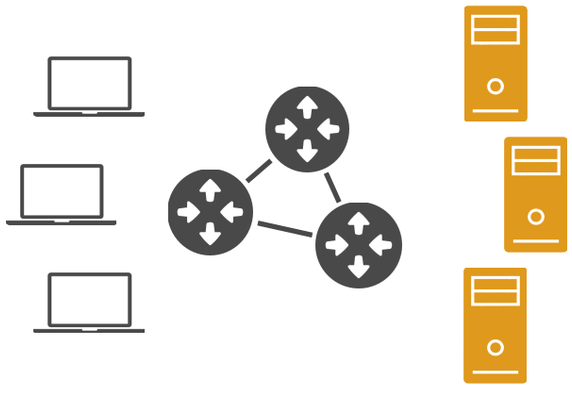
Can the adversary…

…learn sender?



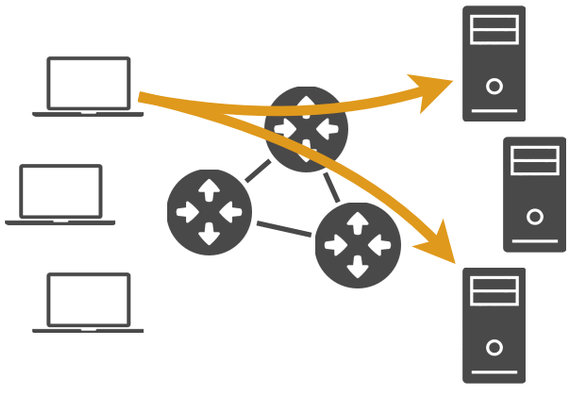
WHO

…learn receiver?



WHAT

…link flows?



HISTORY

GOAL

measure "how private" a network architecture or protocol is

1 What is Privacy?

2 Threat Model

3 "Indicators"

Global, passive adversary.

CAN:

Observe traffic on any link/device.

CANNOT:

Modify, drop, or inject packets.

GOAL

measure "how private" a network architecture or protocol is

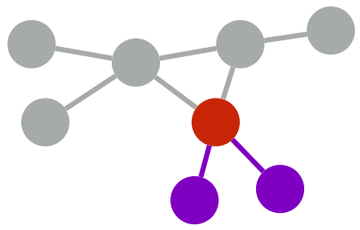
1 What is Privacy?

2 Threat Model

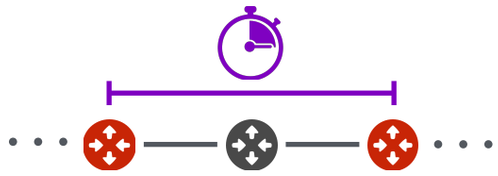
3 "Indicators"

Adversary learns from:

Headers



Topology



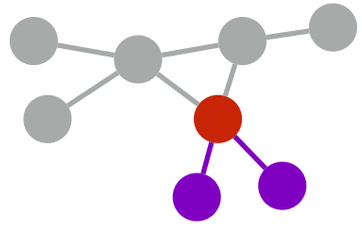
Timing

Choice of indicators has a big impact on measurement tool

Headers

WHAT YOU MEASURE | Properties of an architecture

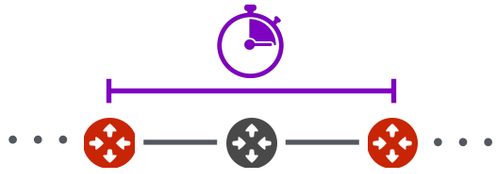
HOW YOU MEASURE | Model headers and devices



Topology

WHAT YOU MEASURE | Properties of a deployed network

HOW YOU MEASURE | Model topology and traffic



Timing

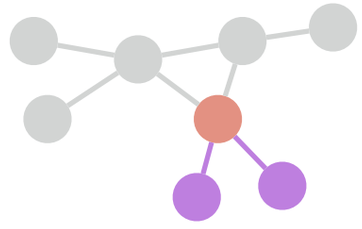
Choice of indicators has a big impact on measurement tool

Headers

WHAT YOU MEASURE | Properties of an architecture

HOW YOU MEASURE | Model headers and devices

SHARE COUNT ANALYSIS

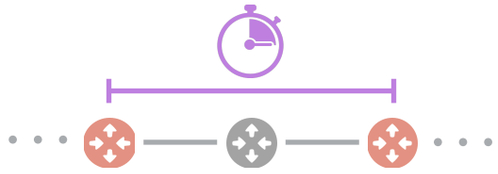


Topology

WHAT YOU MEASURE | Properties of a deployed network

HOW YOU MEASURE | Model topology and traffic

PRIOR WORK



Timing

Share Count Analysis

1

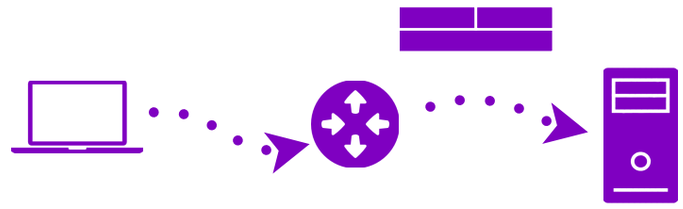
Model header information leakage

2

Model how devices modify headers

3

Measure leakage over test path

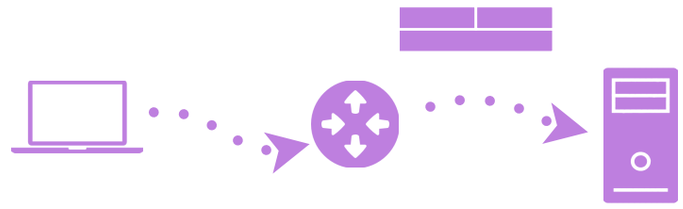


Share Count Analysis

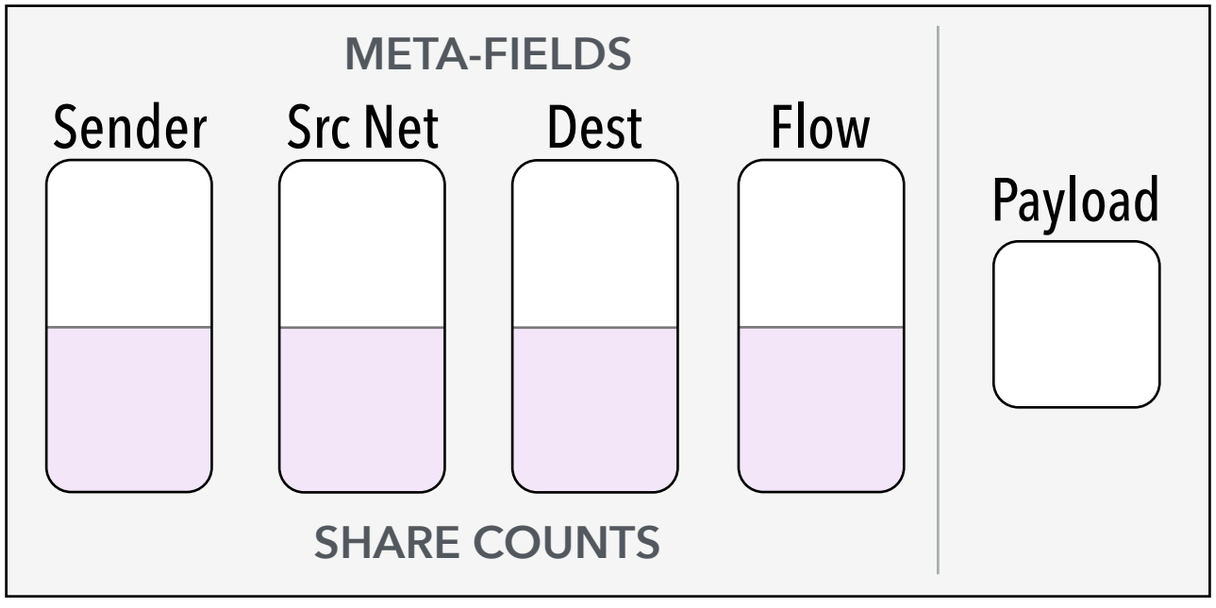
1 Model header information leakage

2 Model how devices modify headers

3 Measure leakage over test path

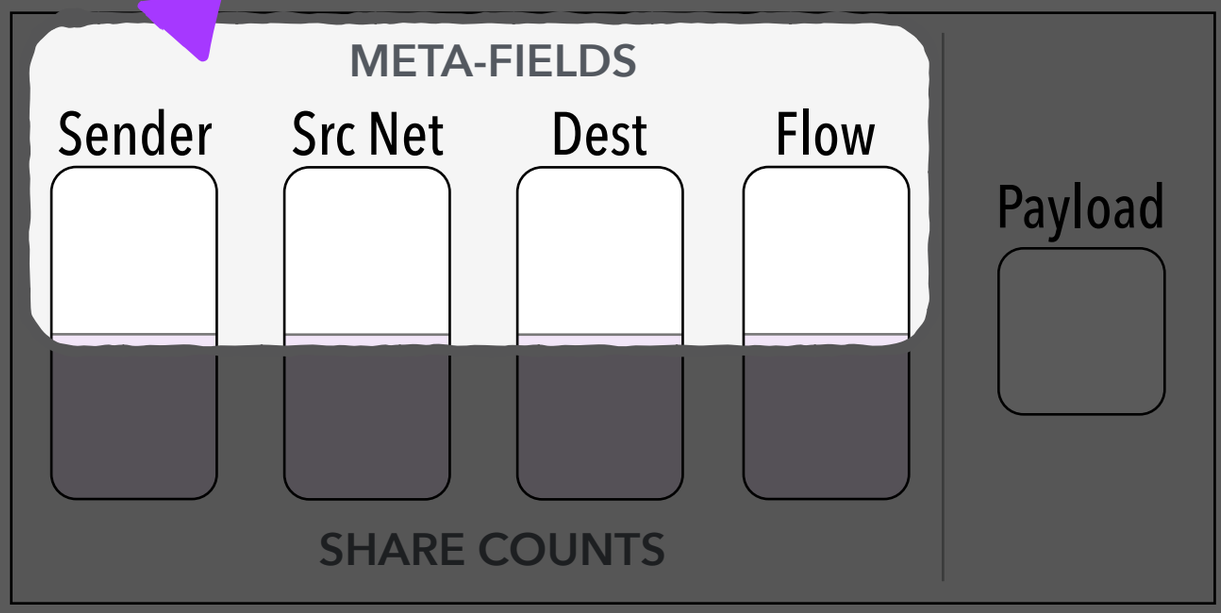


1 Model header information leakage

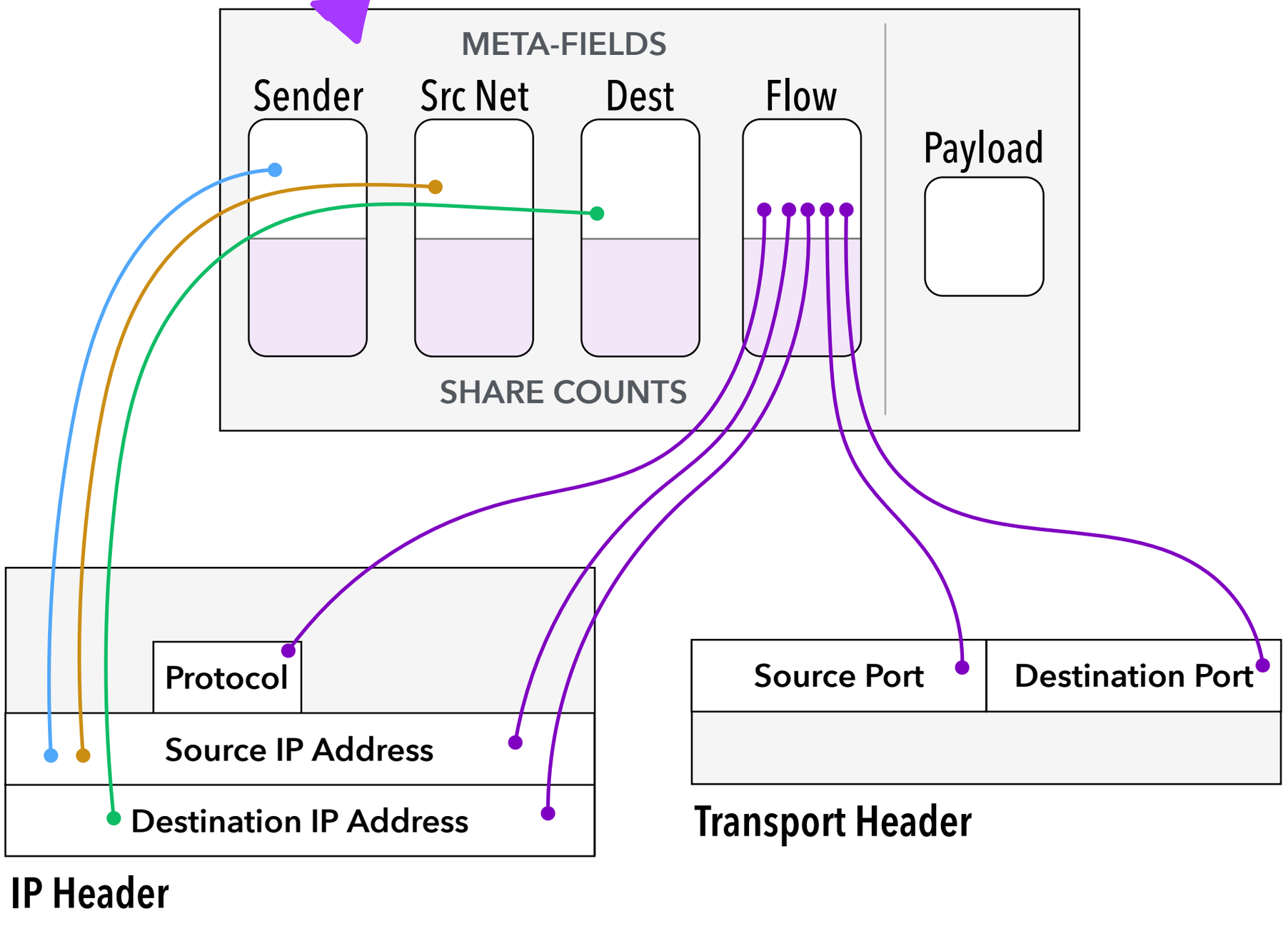


1 Model header information leakage

replace headers with generic, privacy-related *meta-fields*

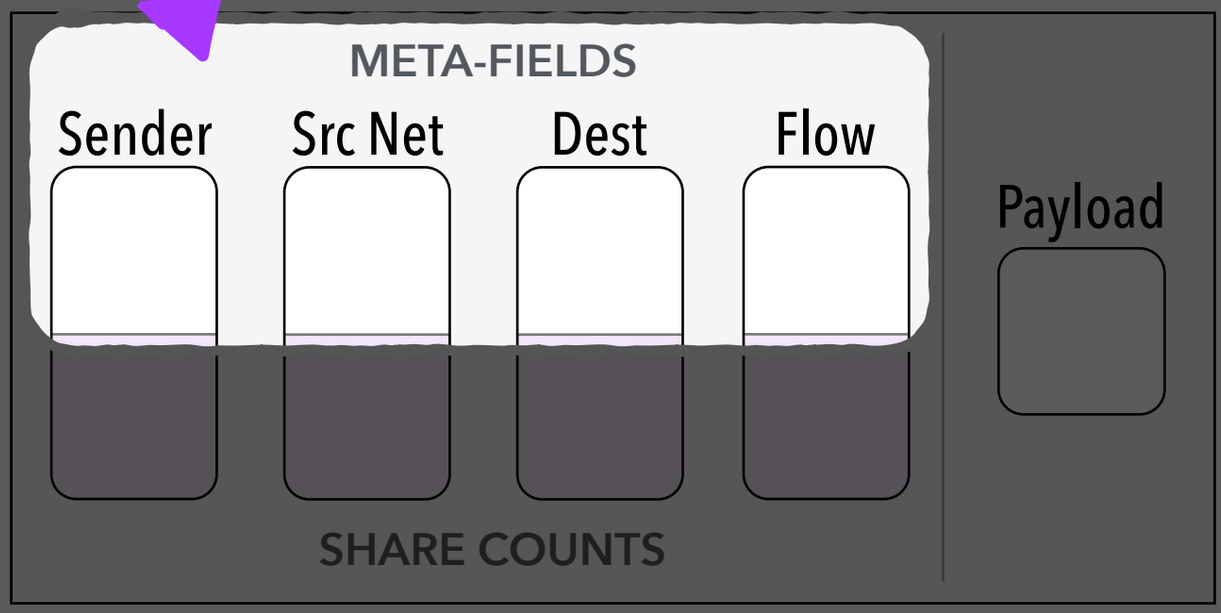


replace headers with generic, privacy-related *meta-fields*



1 Model header information leakage

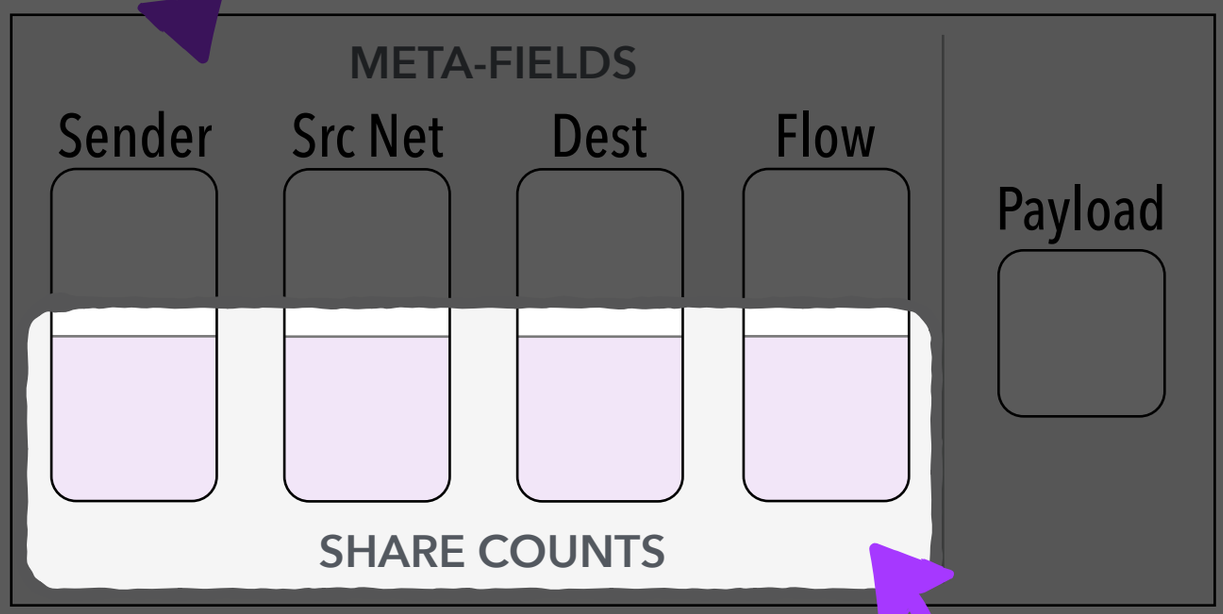
replace headers with generic, privacy-related *meta-fields*



1

Model header information leakage

replace headers with generic, privacy-related *meta-fields*

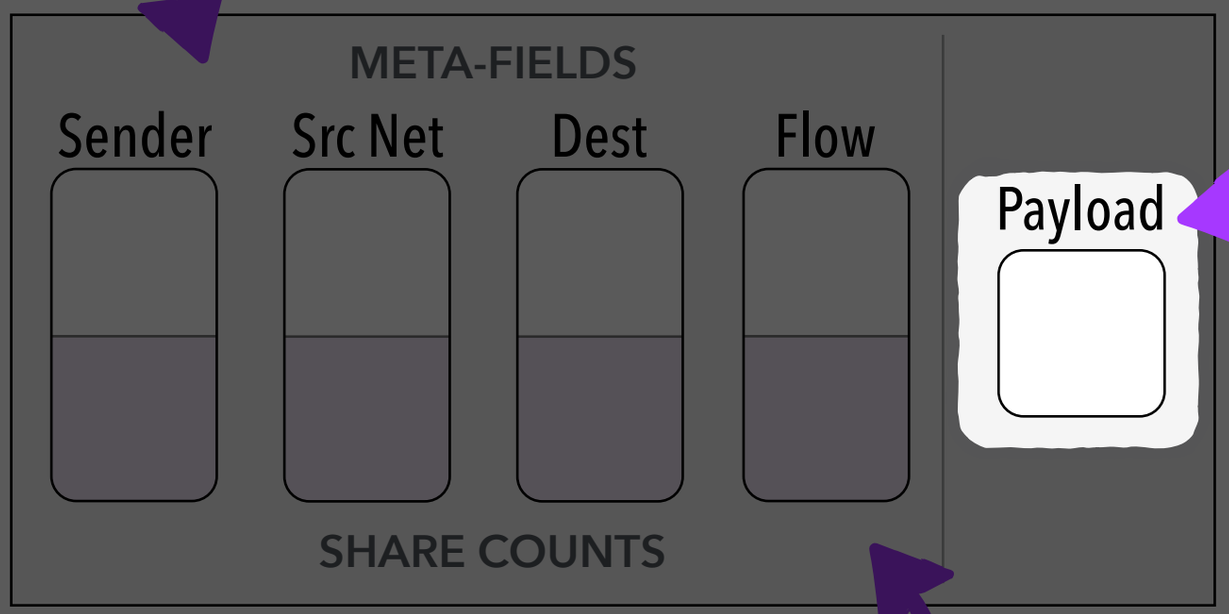


share counts indicate how many entities could share the same value

1

Model header information leakage

replace headers with generic, privacy-related *meta-fields*



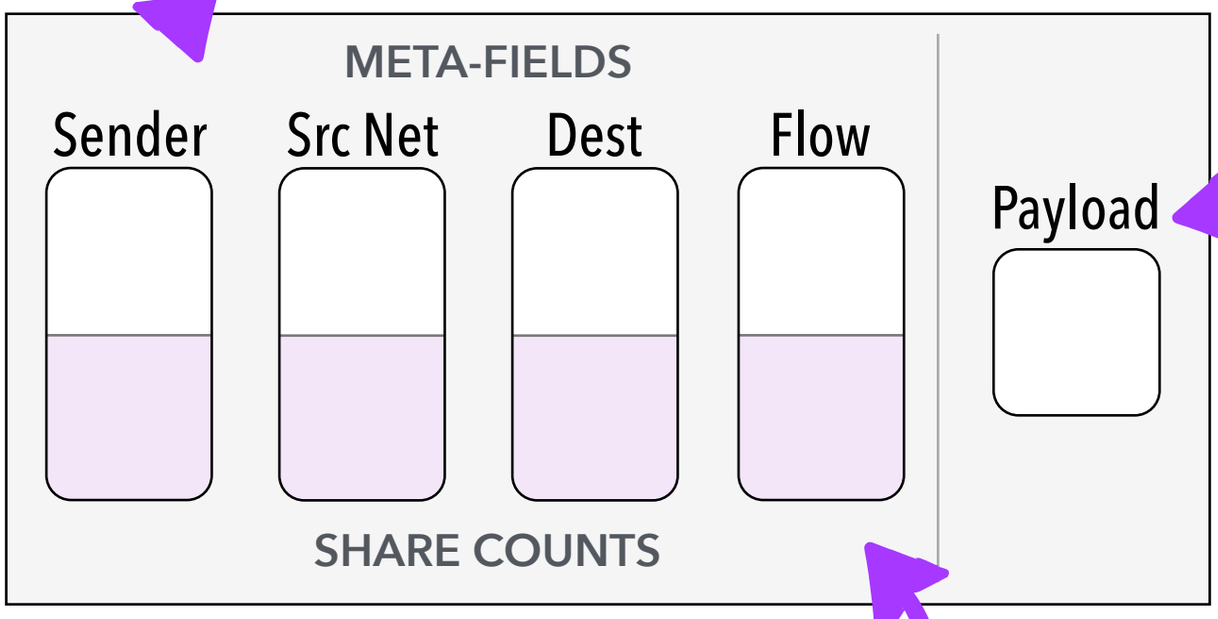
share counts indicate how many entities could share the same value

body is opaque value only used to link multiple sightings of a packet

1

Model header information leakage

replace headers with generic, privacy-related *meta-fields*



share counts indicate how many entities could share the same value

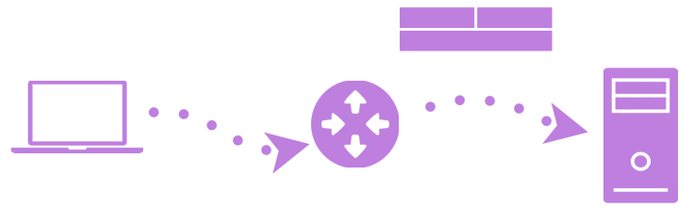
body is opaque value only used to link multiple sightings of a packet

Share Count Analysis

1 Model header information leakage

2 Model how devices modify headers

3 Measure leakage over test path



Share Count Analysis

1

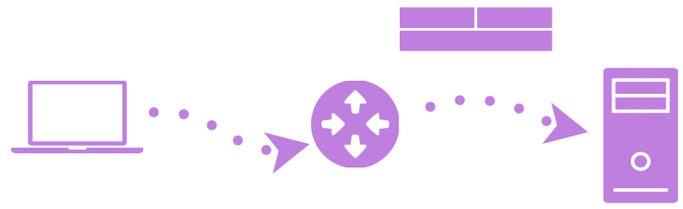
Model header information leakage

2

Model how devices modify headers

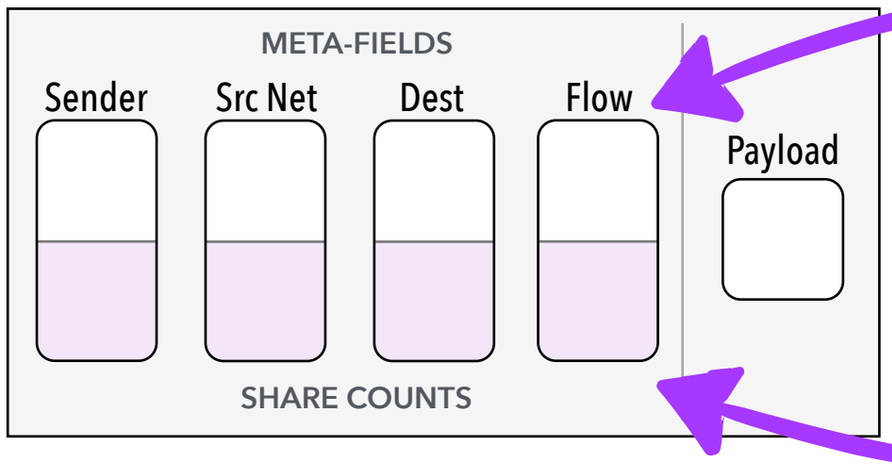
3

Measure leakage over test path



2 Model how devices modify headers

which fields are updated?




```
NAT = {
  update_fields:
    sender,
    flow
  new_share_counts:
    sender: Hn
}
```

what are the new share counts?

H_n = # hosts in source network

Share Count Analysis

1

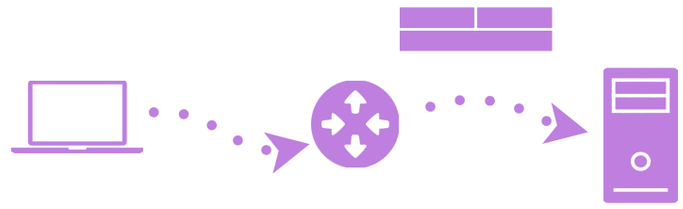
Model header information leakage

2

Model how devices modify headers

3

Measure leakage over test path



Share Count Analysis

1

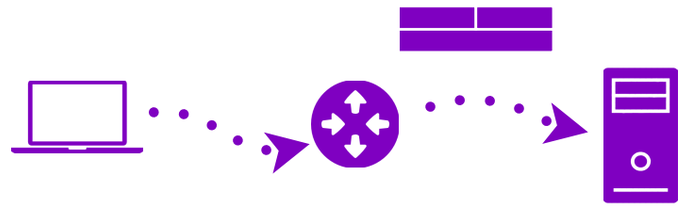
Model header information leakage

2

Model how devices modify headers

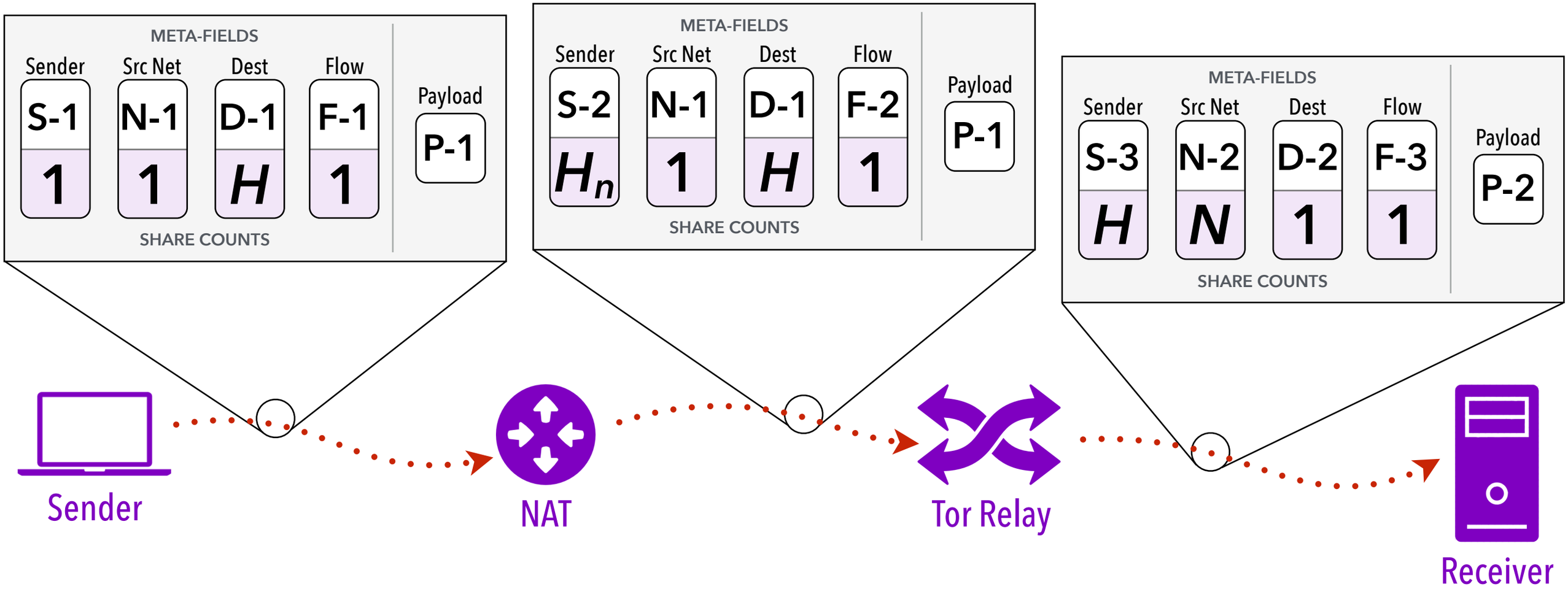
3

Measure leakage over test path

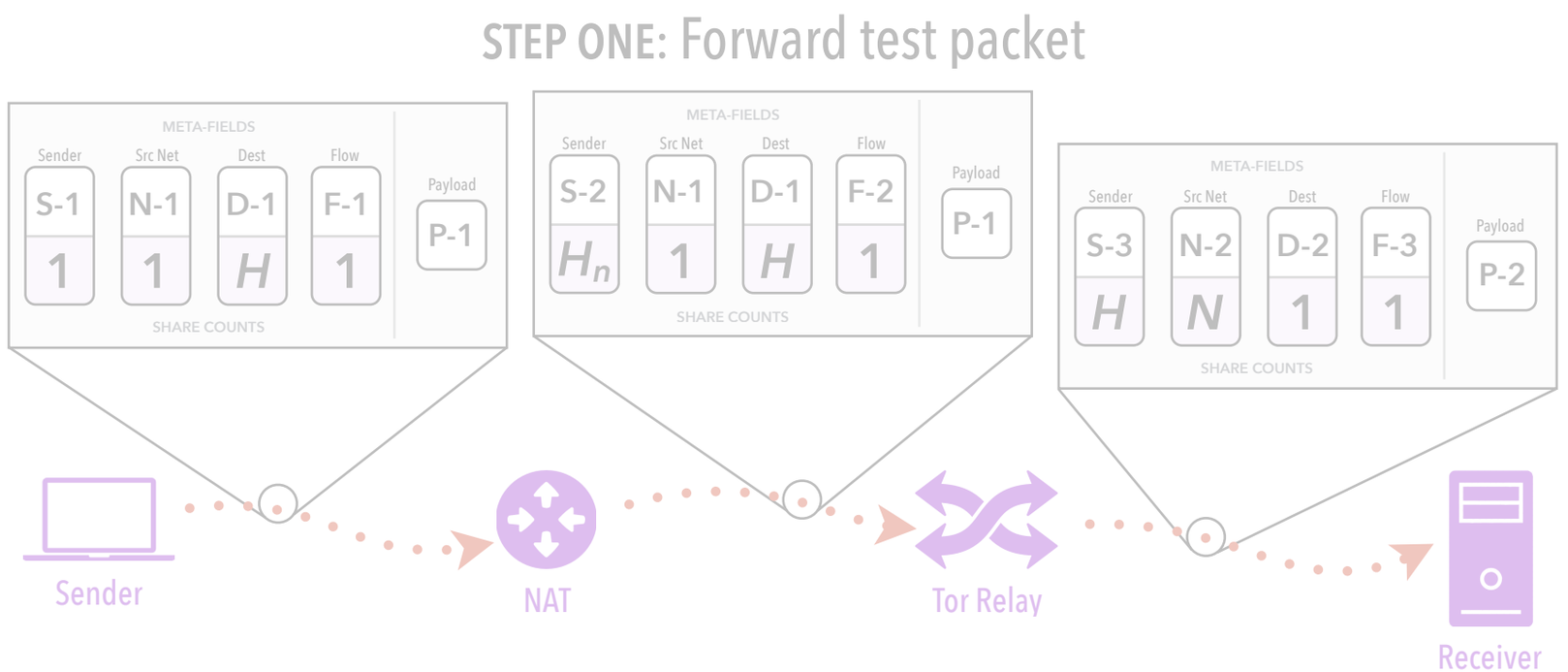


3 Measure leakage over test path

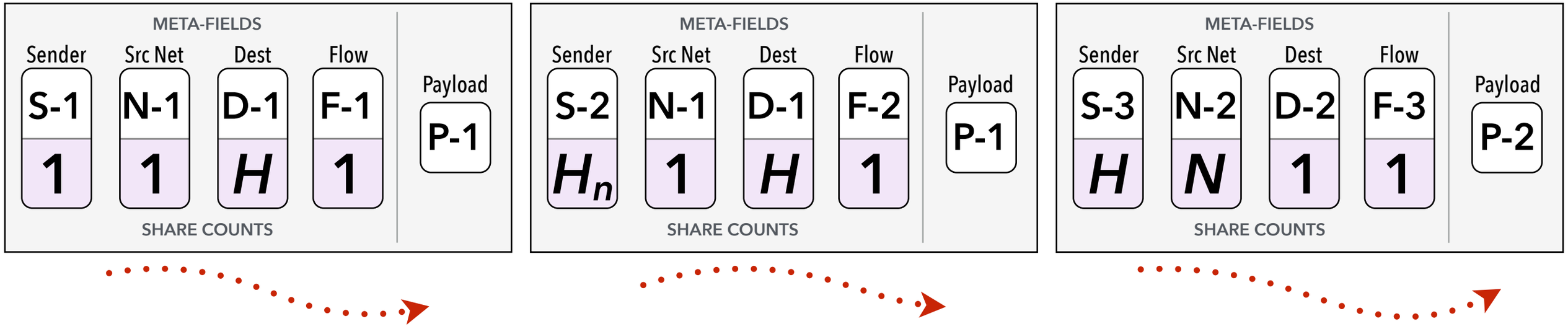
STEP ONE: Forward test packet



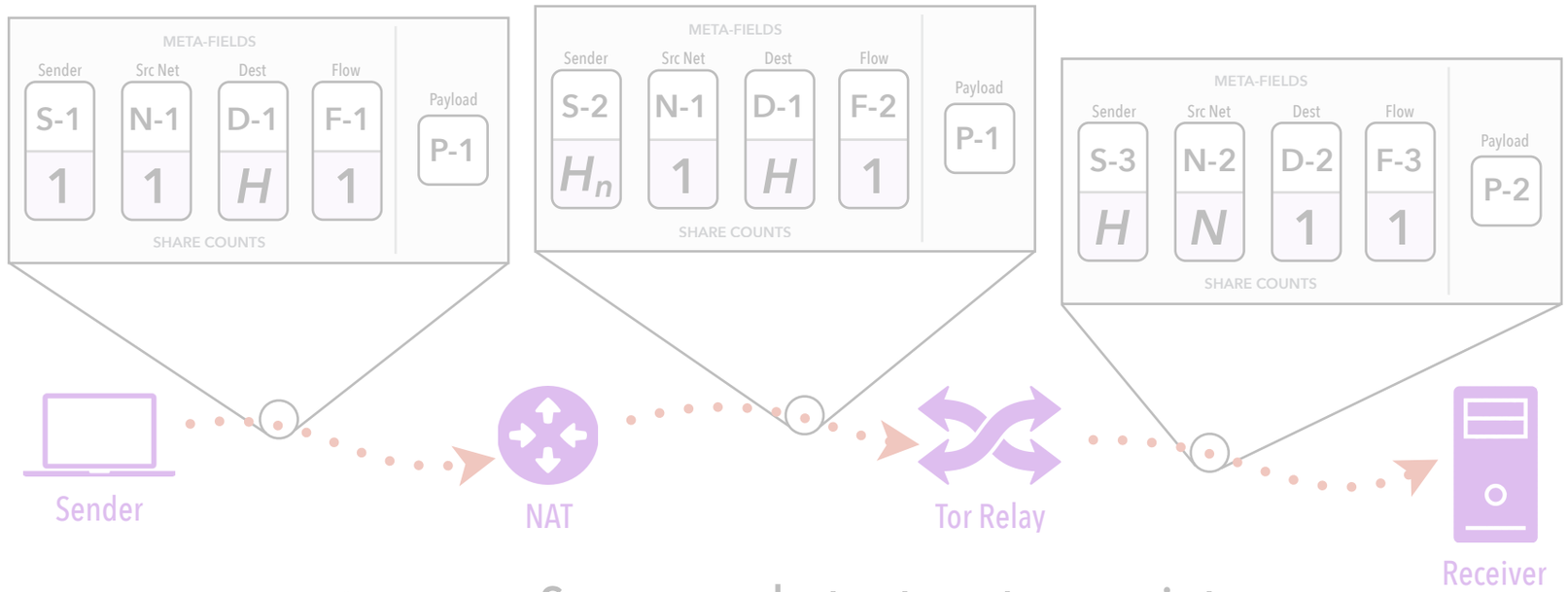
H = # hosts *N* = # networks H_n = # hosts in source network



STEP TWO: Save snapshots at vantage points

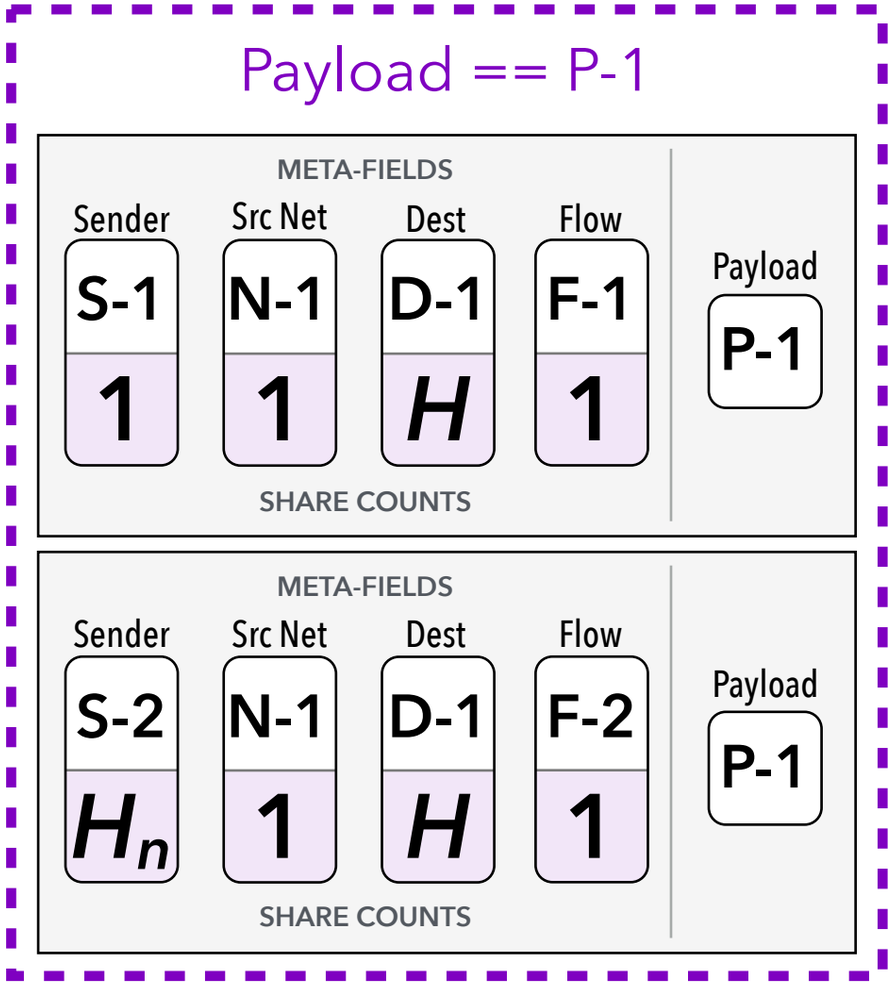


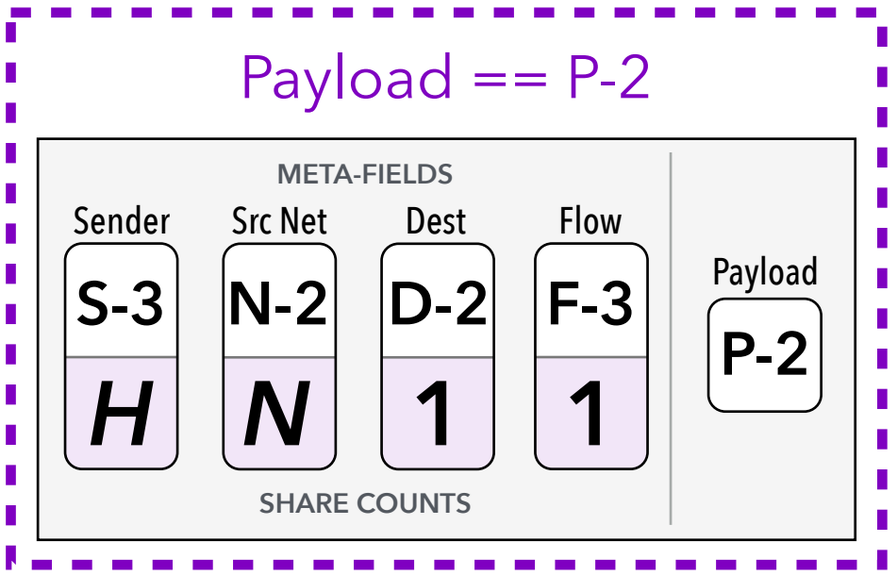
STEP ONE: Forward test packet



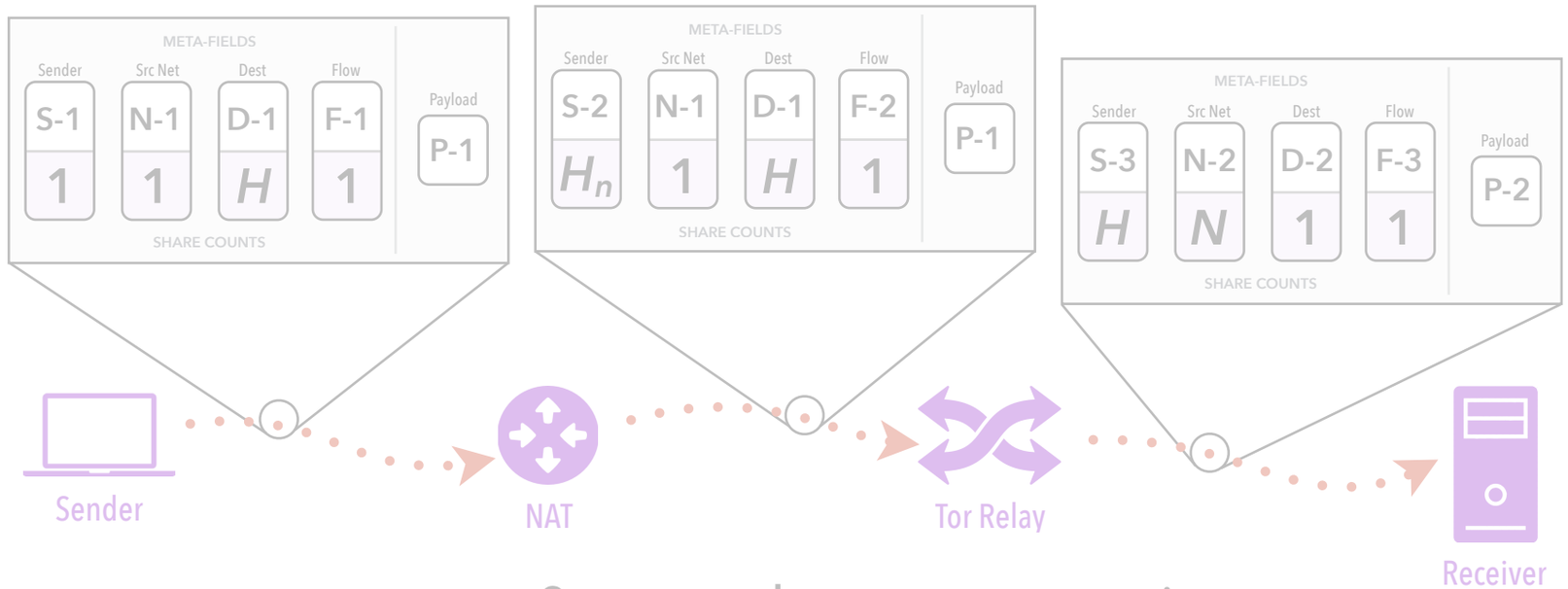
STEP TWO: Save snapshots at vantage points

STEP THREE: Group "linkable" snapshots



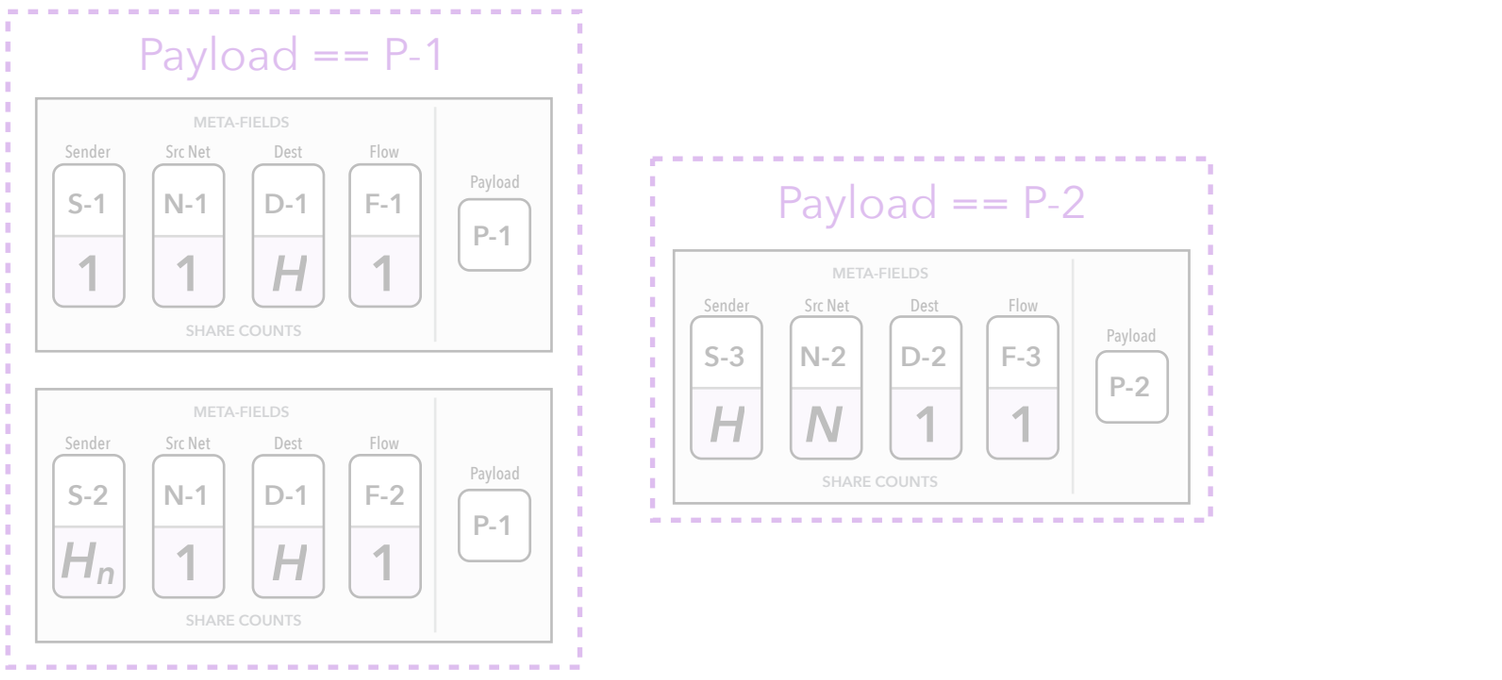


STEP ONE: Forward test packet



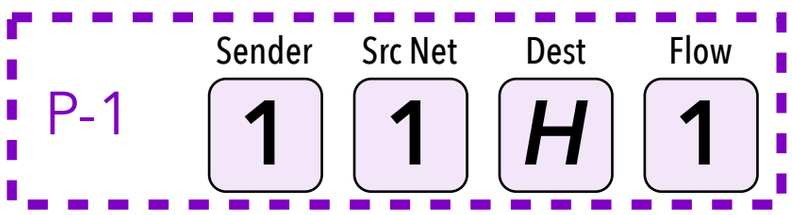
STEP TWO: Save snapshots at vantage points

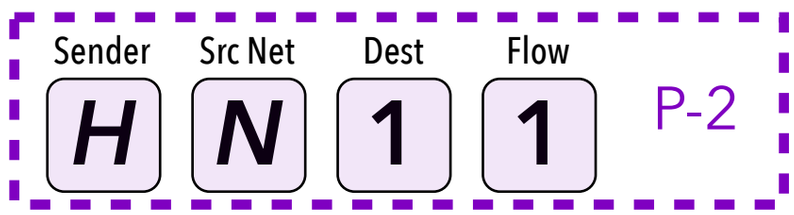
STEP THREE: Group "linkable" snapshots



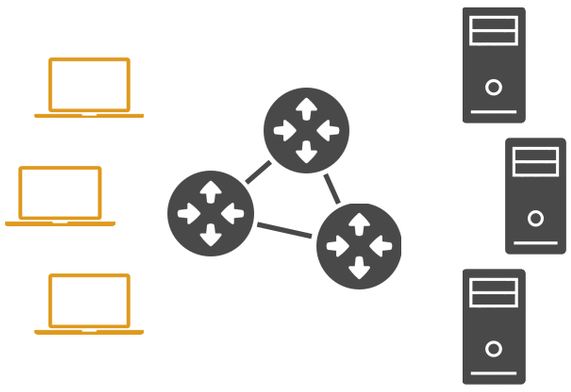
STEP FOUR: Find minimum share counts for each group

Minimum share counts tell us what the adversary learned



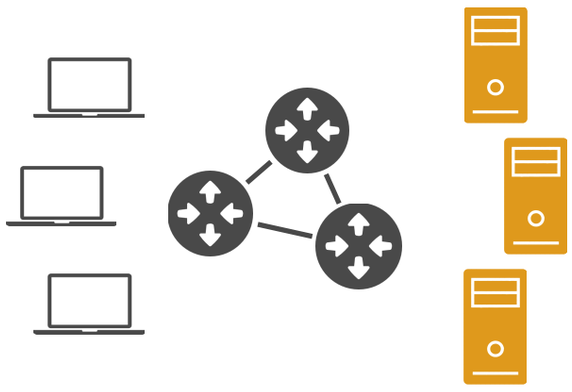


…learn sender?



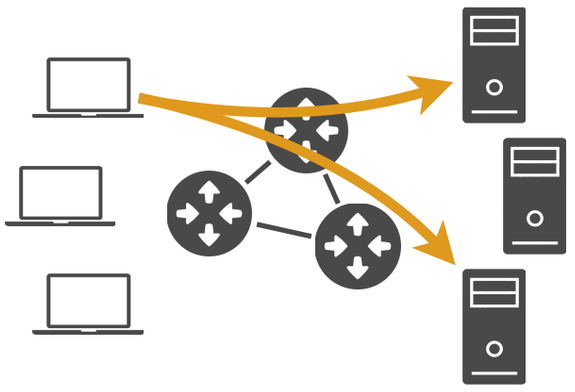
WHO

…learn receiver?



WHAT

…link flows?



HISTORY

Minimum share counts tell us what the adversary learned

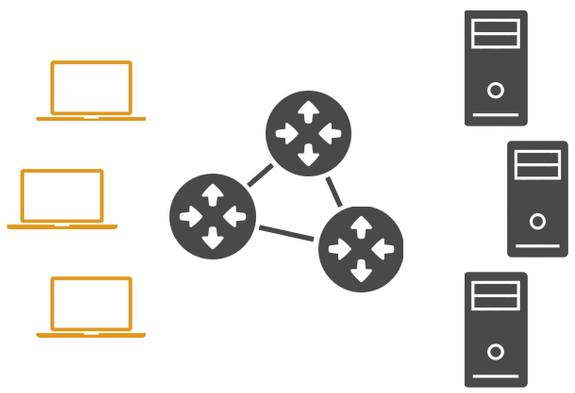
P-1

Sender	Src Net	Dest	Flow
1	1	*H*	1

Sender	Src Net	Dest	Flow
H	*N*	1	1

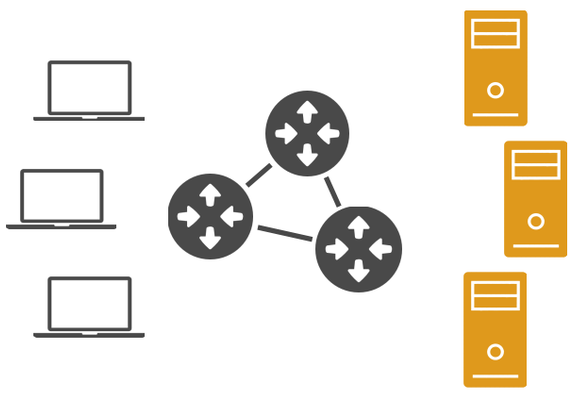
P-2

…learn sender?



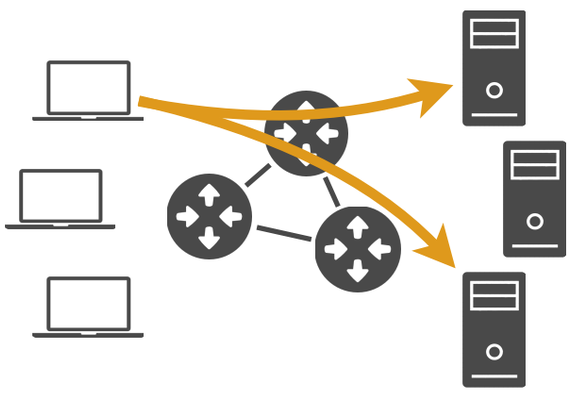
`src net share count == 1`

…learn receiver?



`dest share count == 1`

…link flows?



`sender share count == 1 &&`
`dest share count == 1`

Share Count Analysis

1

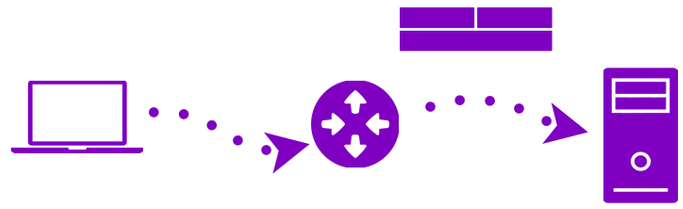
Model header information leakage

2

Model how devices modify headers

3

Measure leakage over test path



Share Count Analysis

1

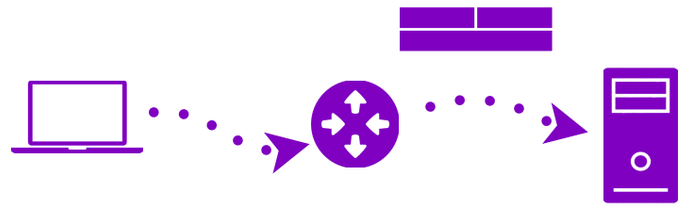
Model header information leakage

2

Model how devices modify headers

3

Measure leakage over test path



Open Questions

1 Automate meta-field and device specs? **Ease of Use**

From traces? From code? From high-level protocol spec?

2 Does our model capture all architectures? **Generality**

Path-based architectures? In-network state (e.g., MPLS, NDN)?

3 Analyze payloads of common protocols **Completeness**

e.g., DHCP, DNS, & TLS handshake

4 Analyze instances of an architecture **Completeness**

Use topology and timing to limit share counts?

It's 11PM.

DO YOU KNOW WHERE YOUR HEADERS ARE?

Measuring Network Privacy with

Share Count Analysis

David Naylor

Peter Steenkiste

Carnegie Mellon University