Drift-Bottle: A Lightweight and Distributed Approach to Failure Localization in General Networks

Xudong Zuo, Qing Li*, Jingyu Xiao, Dan Zhao, Yong Jiang

清華大學深圳国际研究生院
Tsinghua Shenzhen International Graduate School

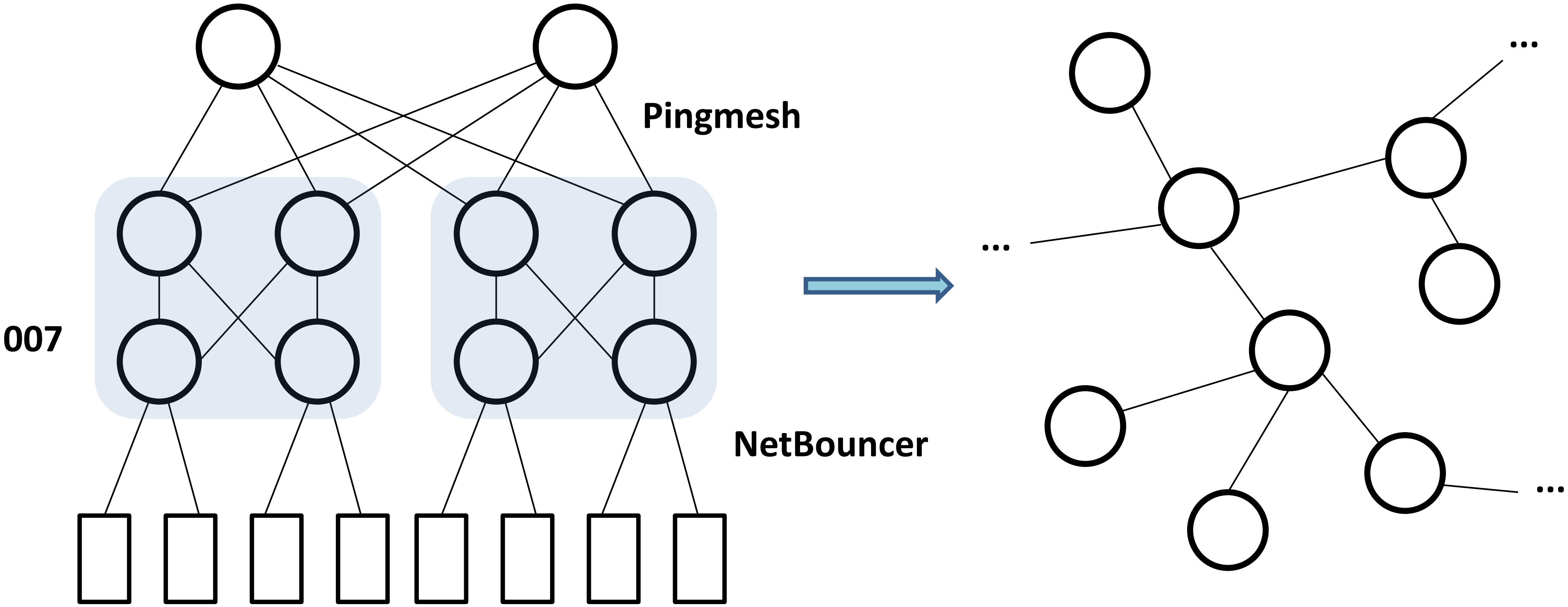
鹏城实验室
PENGCHENG LABORATORY

Failures in Computer Networks

- **Network Failures**: **link failures, link corruptions**, node failures, misconfiguration of flow tables...
- **Harm of Network Failures**: impairs network performance by affecting latency and throughput of data transmission
- It is essential for network operators to detect and localize the failed or corrupted links as quick as they can to mitigate the damage

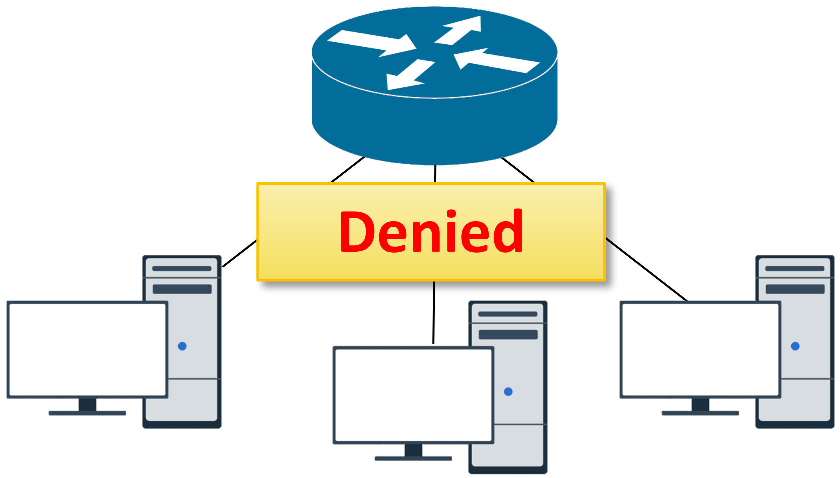
Failure Localization in General Networks

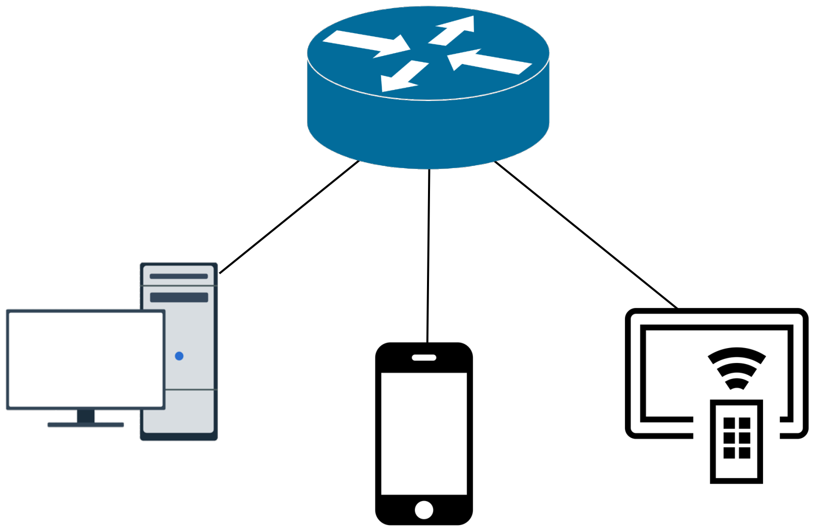
- The topologies of general networks are irregular, which may fail the solutions in DCN



Failure Localization in General Networks

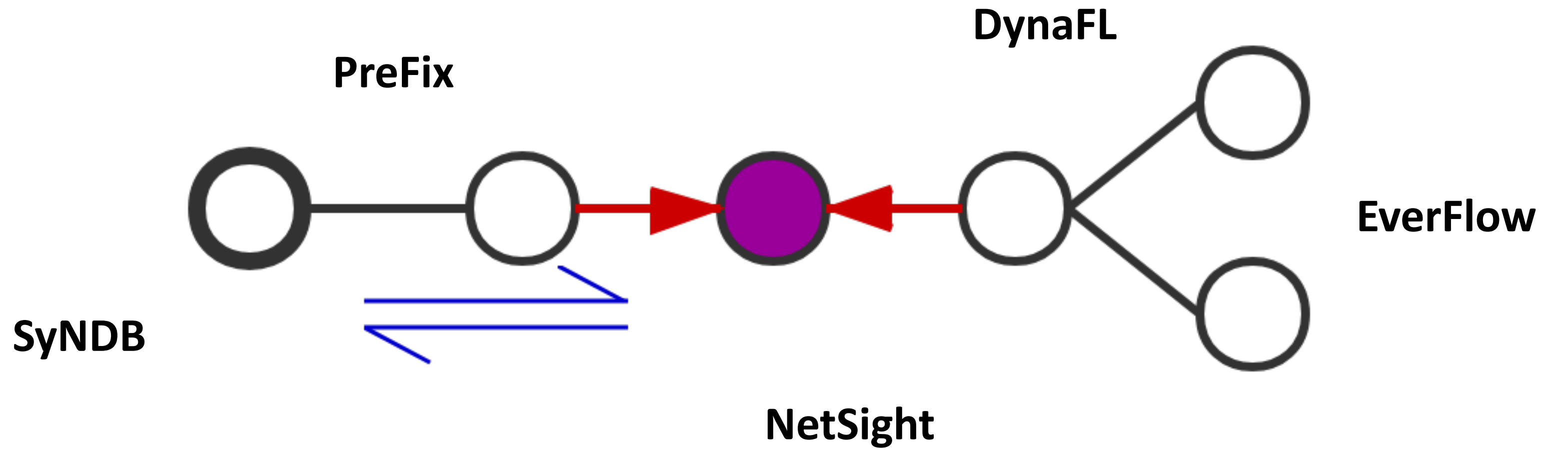
- It is hard to deploy monitoring modules on end hosts





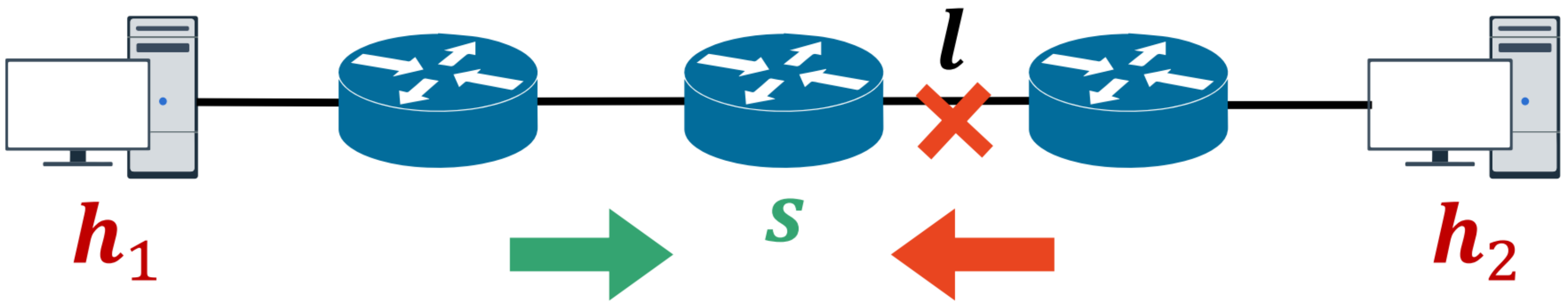
Failure Localization in General Networks

- Existing switch-based solutions may introduce too much overhead to network bandwidth



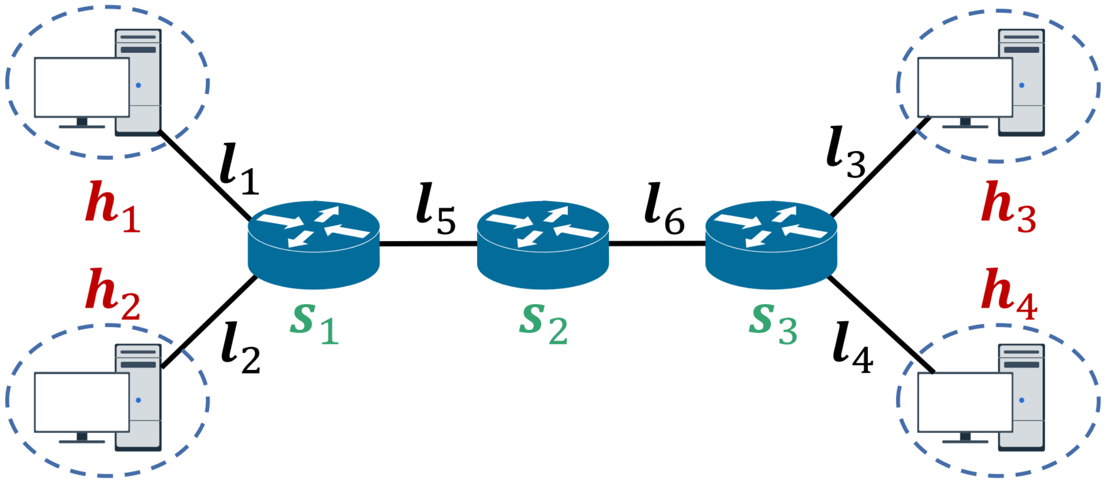
Failure Detection by Flow Monitoring

- Switches can peceive the occurrence of failures by flow monitoring



Failure Localization by Multiple Switches

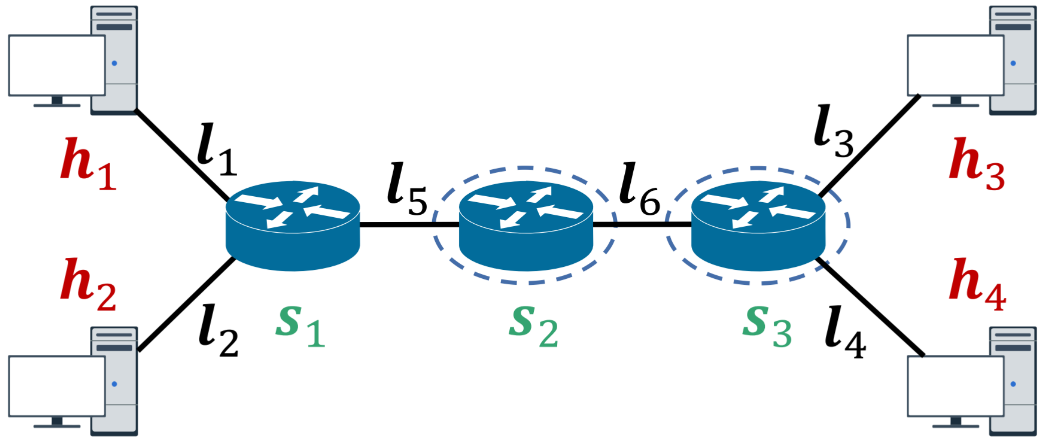
- Multiple switches + multiple flows + data paths = failure location



$$A_{host} = \begin{array}{c} \\ \\ \\ \\ \\ \end{array}\begin{array}{c} \begin{array}{cccccc} l_1 & l_2 & l_3 & l_4 & l_5 & l_6 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} \end{array} \begin{array}{l} \\ h_1 \leftrightarrow h_2 \\ h_1 \leftrightarrow h_3 \\ h_2 \leftrightarrow h_3 \\ h_3 \leftrightarrow h_4 \end{array}$$

Failure Localization by Multiple Switches

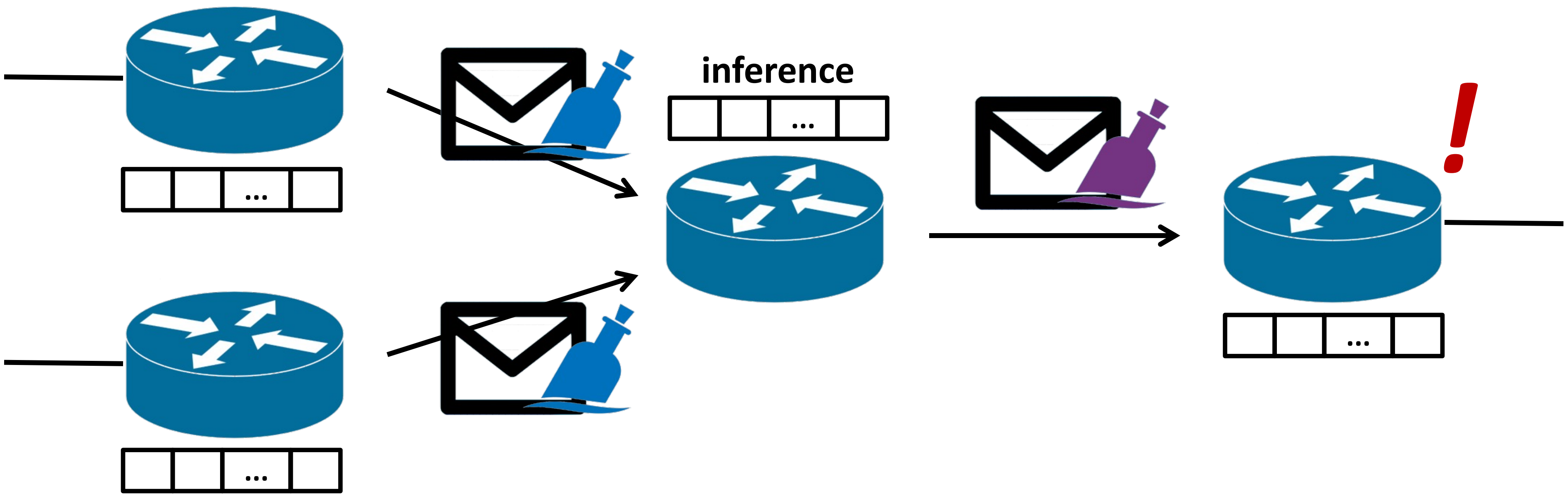
- Multiple switches + multiple flows + data paths = failure location



$$\boldsymbol{A}_{switch} = \begin{array}{c} \begin{array}{cccccc} l_1 & l_2 & l_3 & l_4 & l_5 & l_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array} \begin{array}{l} \\ h_1 \leftrightarrow s_2 \\ h_2 \leftrightarrow s_2 \\ h_3 \leftrightarrow s_2 \\ h_4 \leftrightarrow s_2 \\ h_1 \leftrightarrow s_3 \\ h_4 \leftrightarrow s_3 \end{array}$$

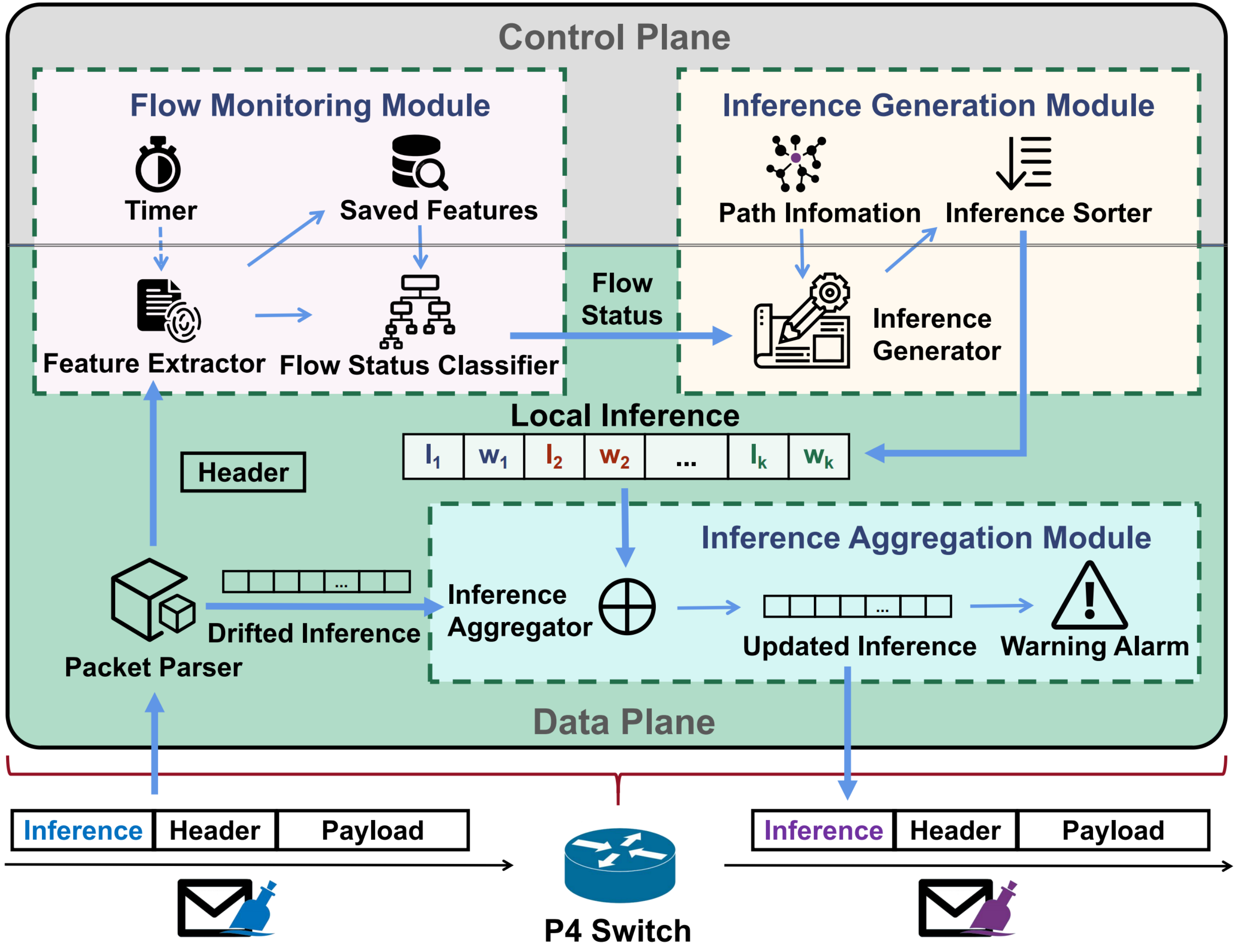
Inference Aggregation by 'Drift'

- Collect the inference along the data path to localize potential failures



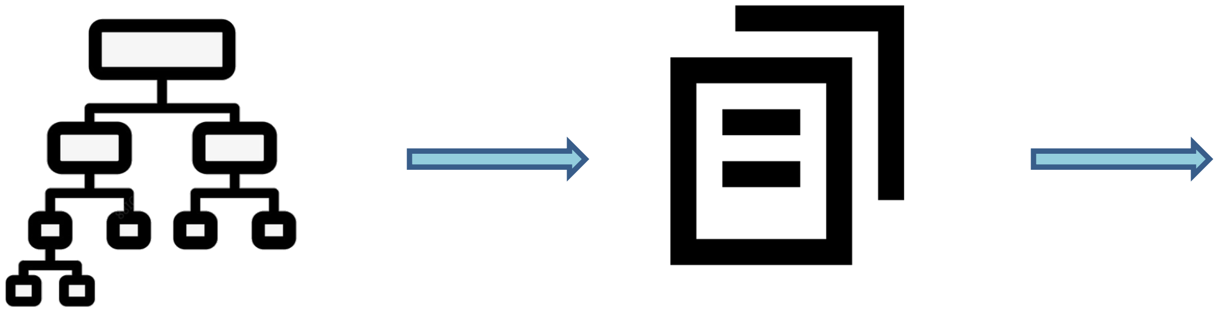
Overview of Drift-Bottle

Introduction
Motivation
Overview
Design
Evaluation



Flow Monitoring Module

- **Goal**: find the flows influenced by potential failures
- **Why Decision Tree**: easy to be transformed into entries of match-action tables on the data plane

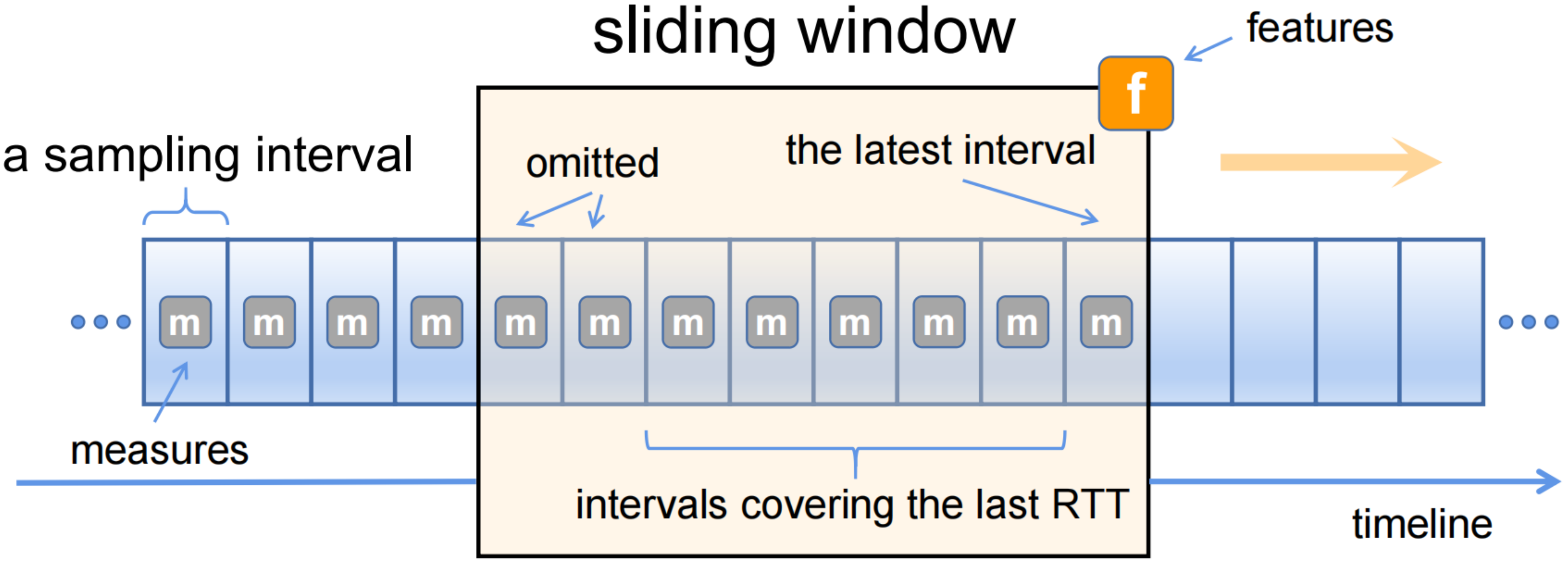


Flow Status:
Normal or
Abnormal

- Operators can customize different flow classifiers

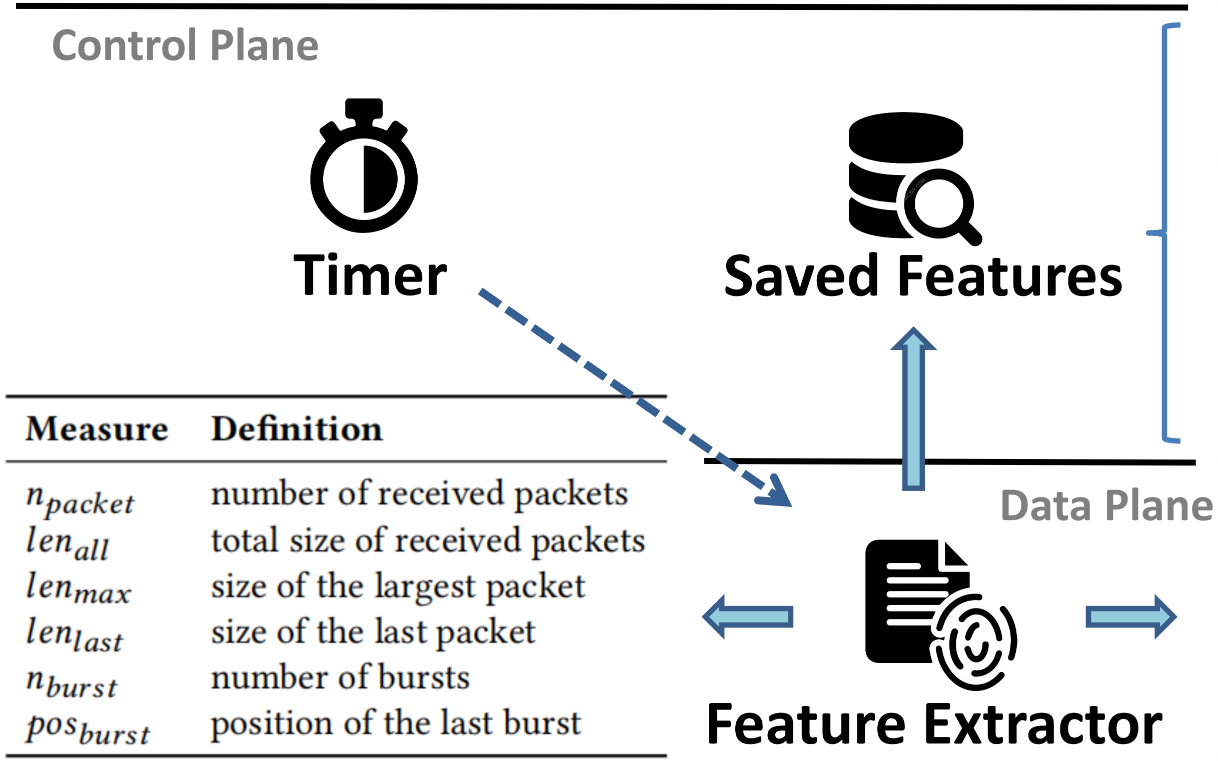
Flow Monitoring Module

- **Measures and Features**



Flow Monitoring Module

- **Definition of measures and features**

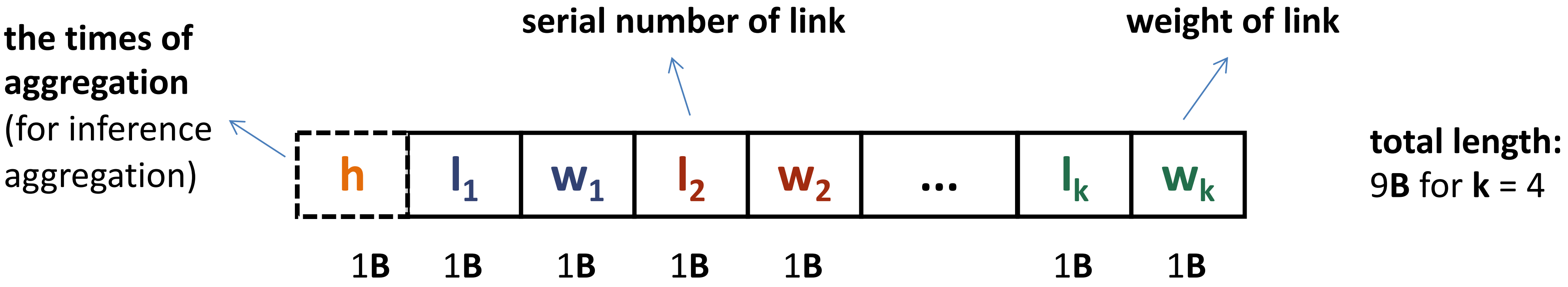


Measure	Definition
n_{packet}	number of received packets
len_{all}	total size of received packets
len_{max}	size of the largest packet
len_{last}	size of the last packet
n_{burst}	number of bursts
pos_{burst}	position of the last burst

Type	Feature	Definition
f_{flow}	RTT	RTT of monitored flow
	len_{path}	length of data path of flow
	$n_{interval}$	n of intervals to cover a RTT
f_{avg}	avg_n_{packet}	avg. n_{packet} of intervals in last RTT
	avg_len_{all}	avg. len_{all} of intervals in last RTT
	avg_len_{max}	avg. len_{max} of intervals in last RTT
	avg_len_{last}	avg. len_{last} of intervals in last RTT
	avg_n_{burst}	avg. n_{burst} of intervals in last RTT
	avg_pos_{burst}	avg. pos_{burst} of intervals in last RTT
f_{last}	$last\_n_{packet}$	n_{packet} in last interval
	$last\_len_{all}$	len_{all} in last interval
	$last\_len_{max}$	len_{max} in last interval
	$last\_len_{last}$	len_{last} in last interval
	$last\_n_{burst}$	n_{burst} in last interval
	$last\_pos_{burst}$	pos_{burst} in last interval

Inference Generation Module

- **Goal**: generate the local inference of potential failures with abnormal flows and their data paths
- **Inference Format**:



Inference Generation Module

- **Weight assignment scheme without the information from normal flows**

$h_1 \rightarrow h_9$ √→×

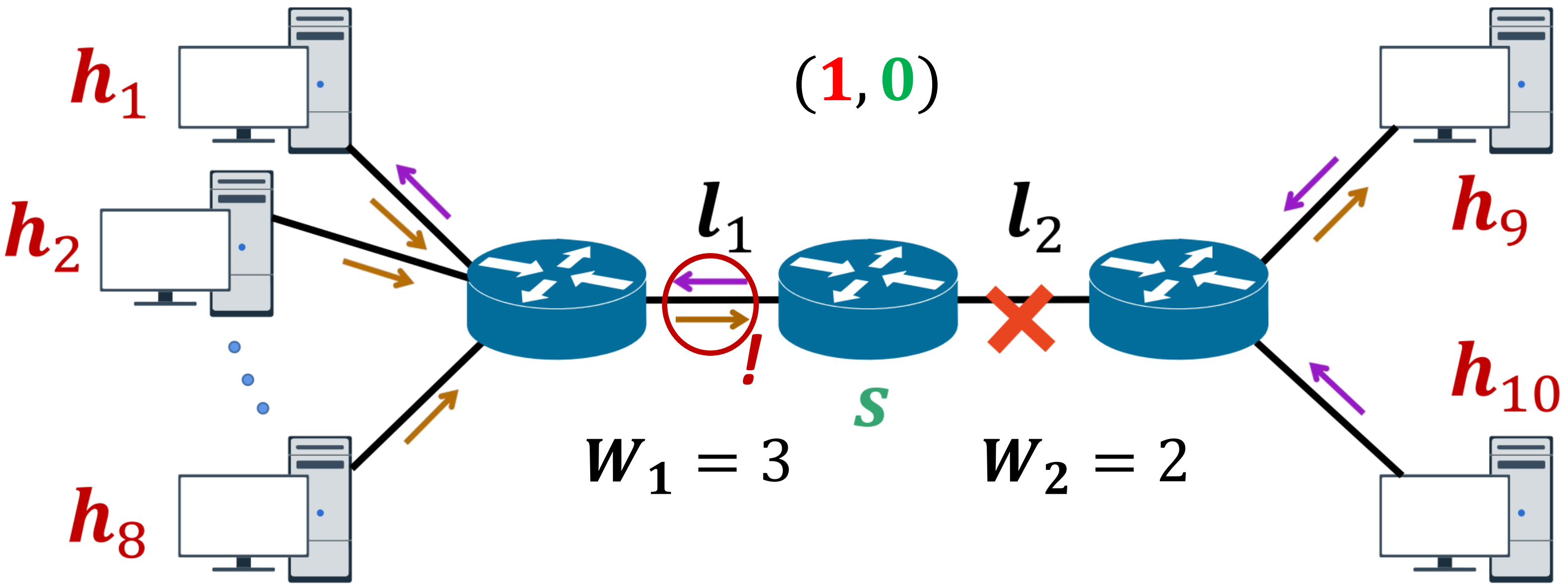
$h_2 \rightarrow h_9$ √→×

$h_3 \rightarrow h_9$ √→×

$h_4 \rightarrow h_9$ √

…

$h_8 \rightarrow h_9$ √



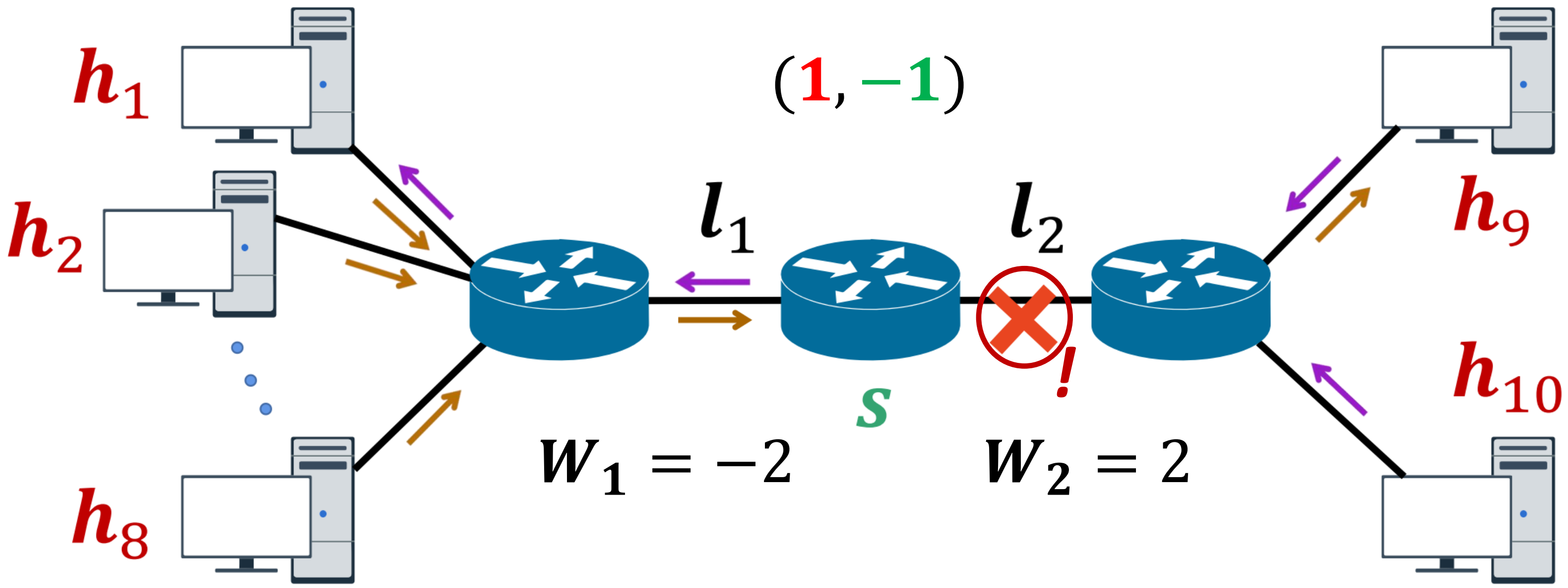
$h_9 \rightarrow h_1$ ×

$h_{10} \rightarrow h_1$ ×

Inference Generation Module

- Weight assignment scheme with the information from normal flows

$h_1 \rightarrow h_9$ √→×
$h_2 \rightarrow h_9$ √→×
$h_3 \rightarrow h_9$ √→×
$h_4 \rightarrow h_9$ √
…
$h_8 \rightarrow h_9$ √



$h_9 \rightarrow h_1$ ×
$h_{10} \rightarrow h_1$ ×

Inference Generation Module

- **Algorithm**

Algorithm 1: Local Inference Generation

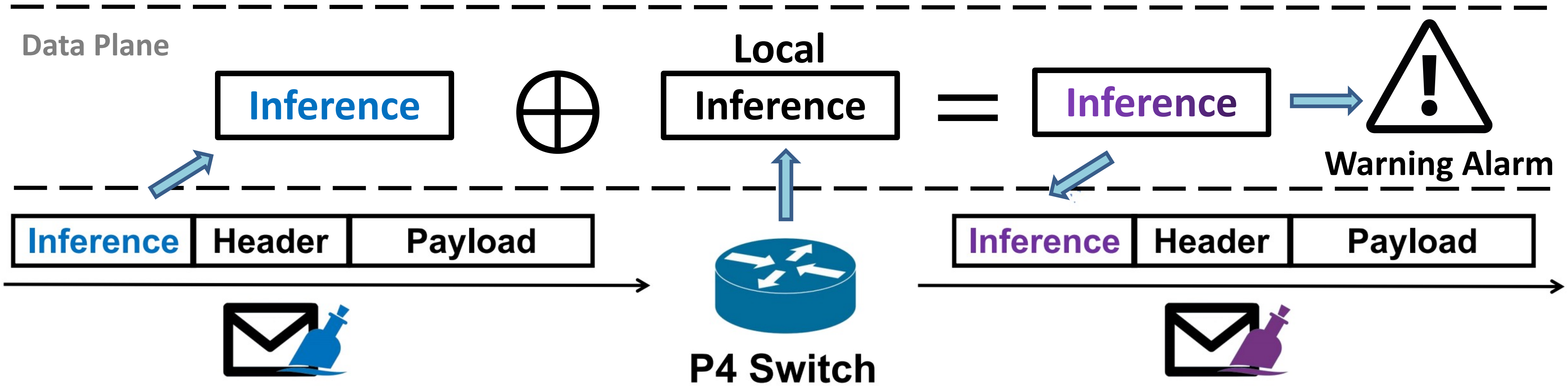
Input: F - set of monitored flows, P - upstream data paths of flows, L - set of links, S - status of monitored flows, k - length of inference
Output: I - local inference about failures

$IF \leftarrow \emptyset$ on the data plane;
for $f \in F$ **do**
  $path_f \leftarrow$ upstream data path of f from P;
  $status_f \leftarrow$ status of f from S;
  if $status_f = abnormal$ **then**
    $I_f \leftarrow \{(l_i, 1) \mid \forall l_i \in path_f\}$;
  else
    $I_f \leftarrow \{(l_i, -1) \mid \forall l_i \in path_f\}$;
  end if
  $IF \leftarrow IF \cup \{I_f\}$;
end for
Upload IF to the control plane;
$I \leftarrow \{(l_i, 0) \mid \forall l_i \in L\}$ on the control plane;
for $I_f \in IF$ **do**
  $I \leftarrow I \bigoplus I_f$;
end for
Remove (l_i, w_i) from I **if** $w_i = 0$;
Sort $I = (l_i, w_i)$ in descending order by w_i;
Truncate I to the k-th (l_i, w_i);
Send I to the data plane;
return $I = \{(l_i, w_i)\}$

Aggregation Operator $\bigoplus$: adds the weight of the same links from two inferences, maintains the others

Inference Aggregation Module

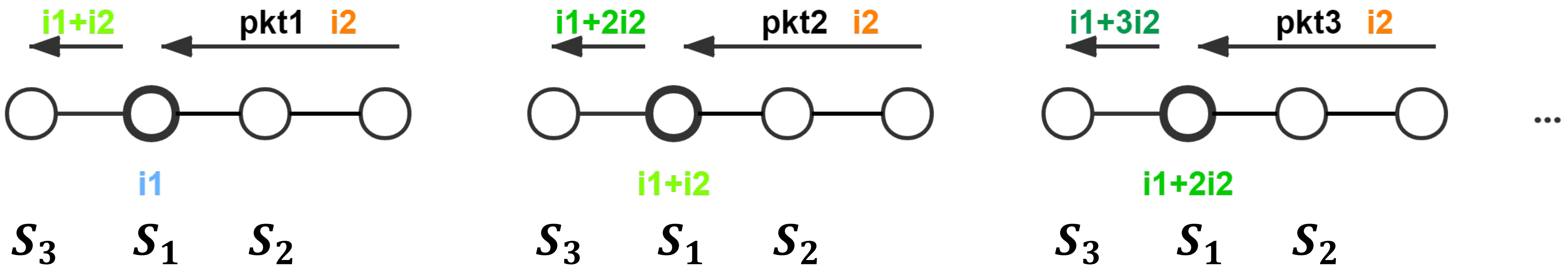
- **Goal**: uses normal packets in the network to aggregate inferences from different switches
- **Inference Processing Logic**:



Inference Aggregation Module

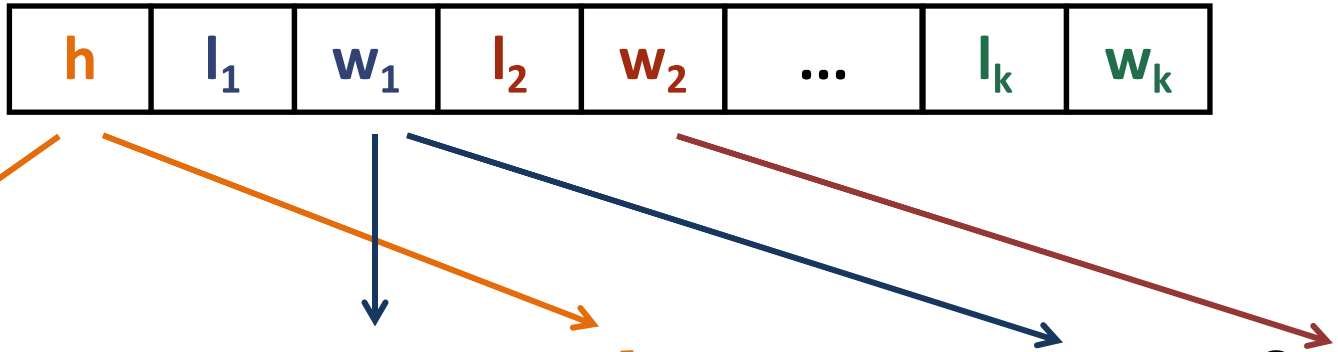
- Switch keeps its local inference unchanged in order to avoid over aggregation

- Over Aggregation:



Inference Aggregation Module

- **Warning Raising Mechanism**



$$hop_{now} \geq hop_{min}$$

condition 1

$$w_1 \geq \alpha * hop_{now}$$

condition 2

$$w_1 \geq \beta * w_2$$

condition 3

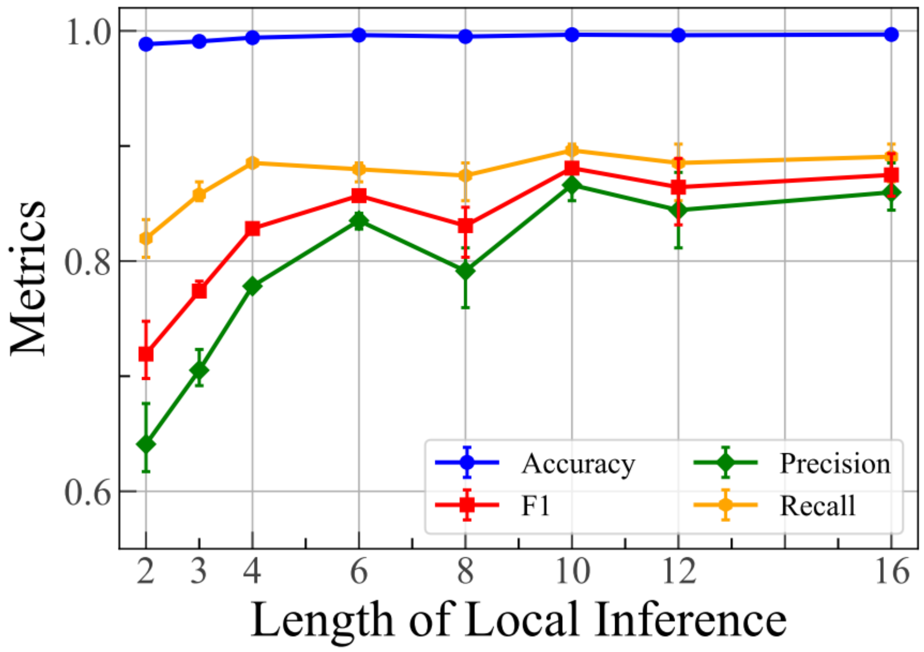
hop_{min} and α are preset thresholds related to the scale of the network
The selection of β is irrelevant to the topology. Read our paper for more details

Evaluation Setup

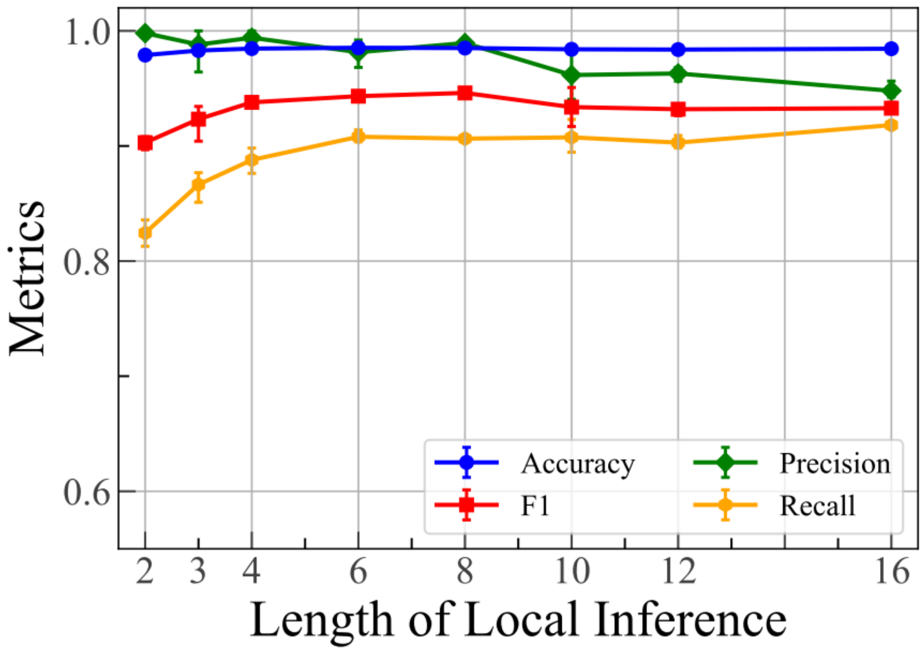
- Simulation by Mininet on 4 chosen topologies
- Generate random traffic with the injection of link failures and corruptions
- Statistics of the chosen topologies:

Topology	Node	Link	VAR. of link latency
Geant2012	40	61	14.12
Chinanet	42	66	8.09
Tinet	53	89	247.64
AS1221	104	151	9.39

Length of Inference



Geant2012



Chinanet

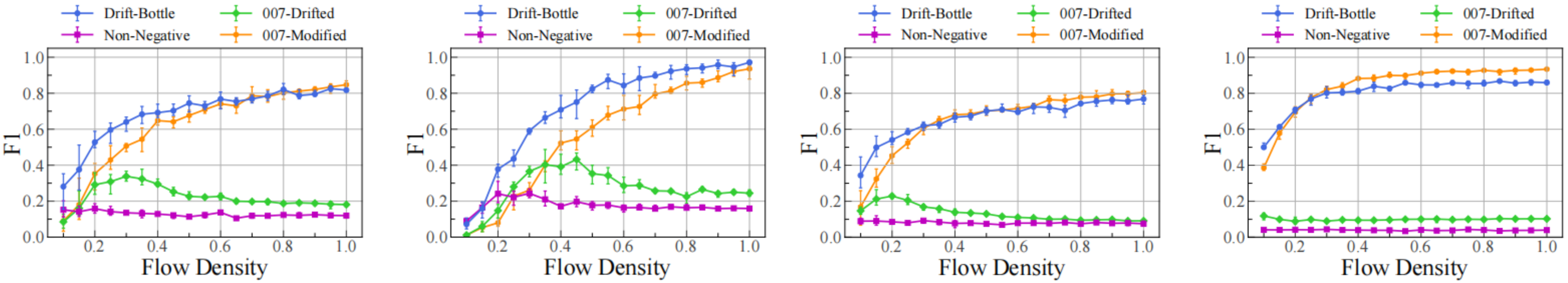
Weight Assignment Scheme

Drift-Bottle: $(\mathbf{1}, -\mathbf{1})$

Non-Negative: $(\mathbf{1}, \mathbf{0})$

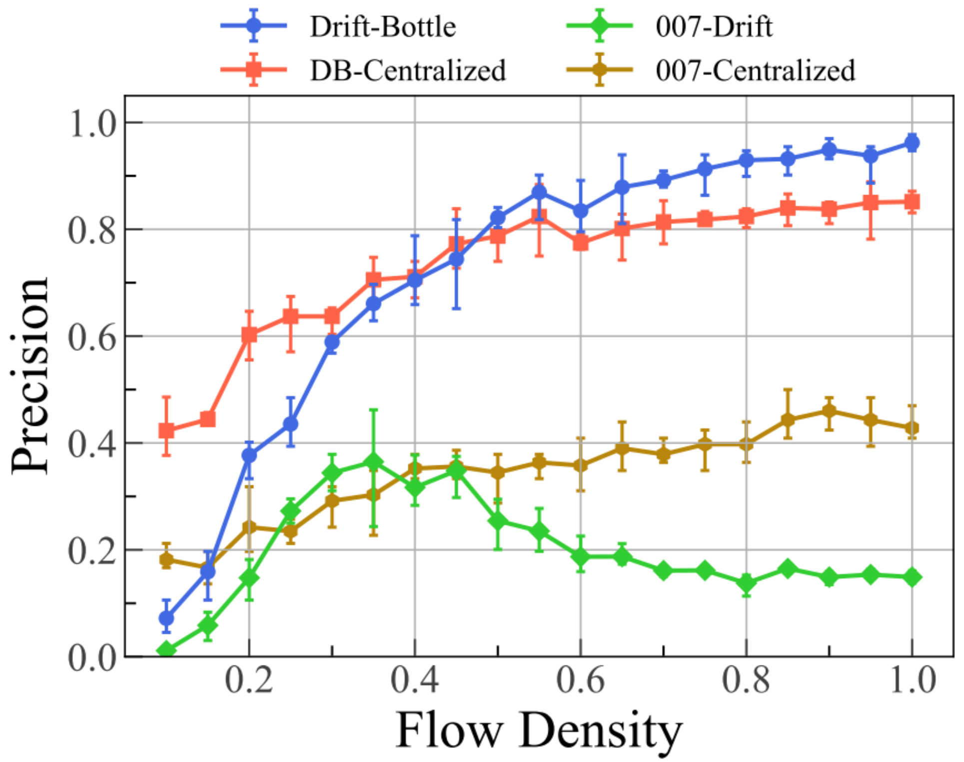
007-Drifted: $(\mathbf{1}/\boldsymbol{n}, \mathbf{0})$

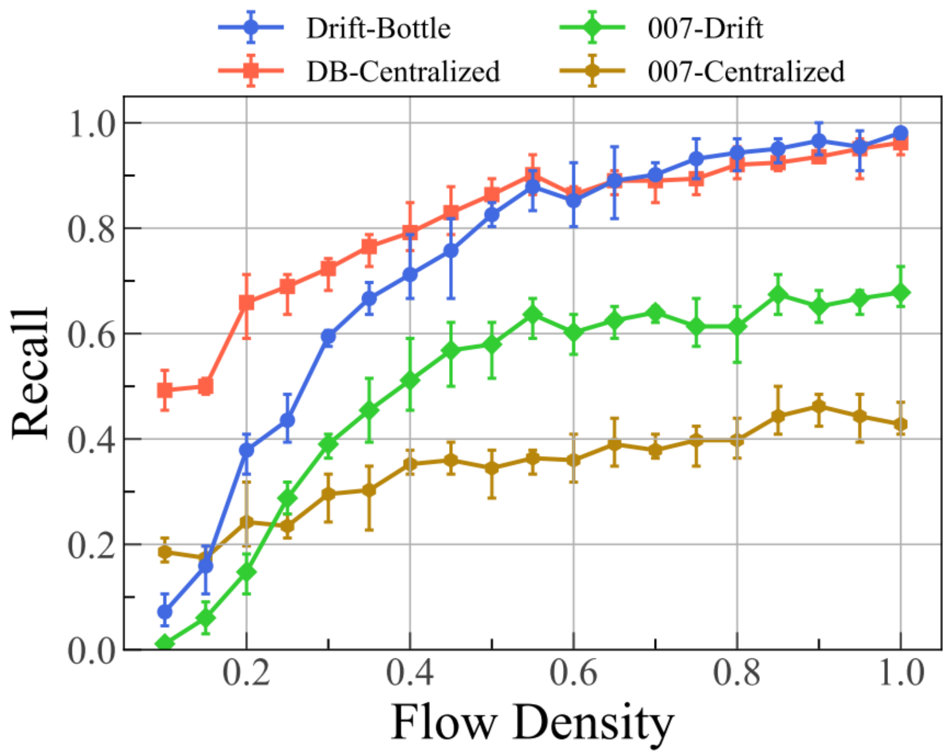
007-Modified: $(\mathbf{1}/\boldsymbol{n}, -\mathbf{1}/\boldsymbol{n})$



(a) Geant2012 (b) Chinanet (c) Tinet (d) AS1221

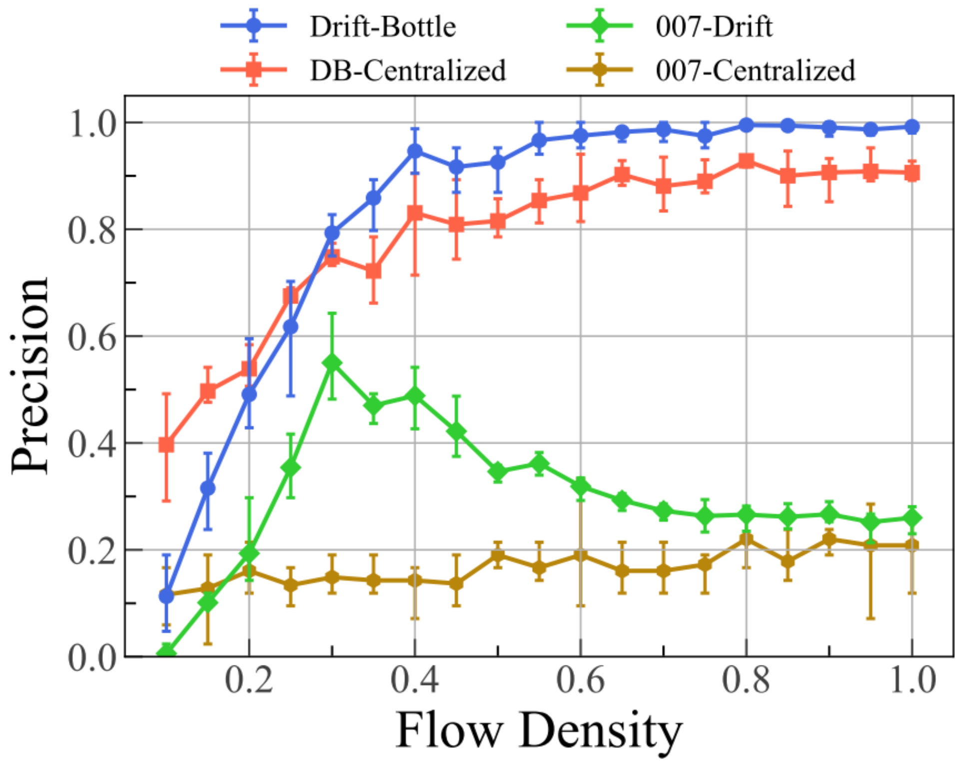
Single Failure Scenario (Chinanet)

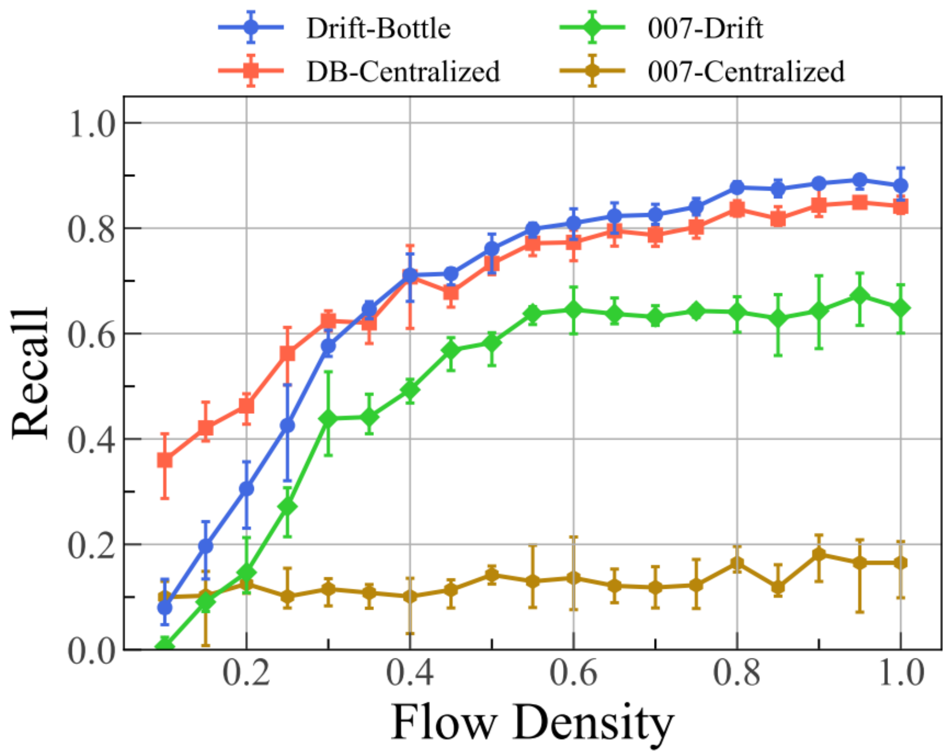




Multiple Failures Scenario (Chinanet)

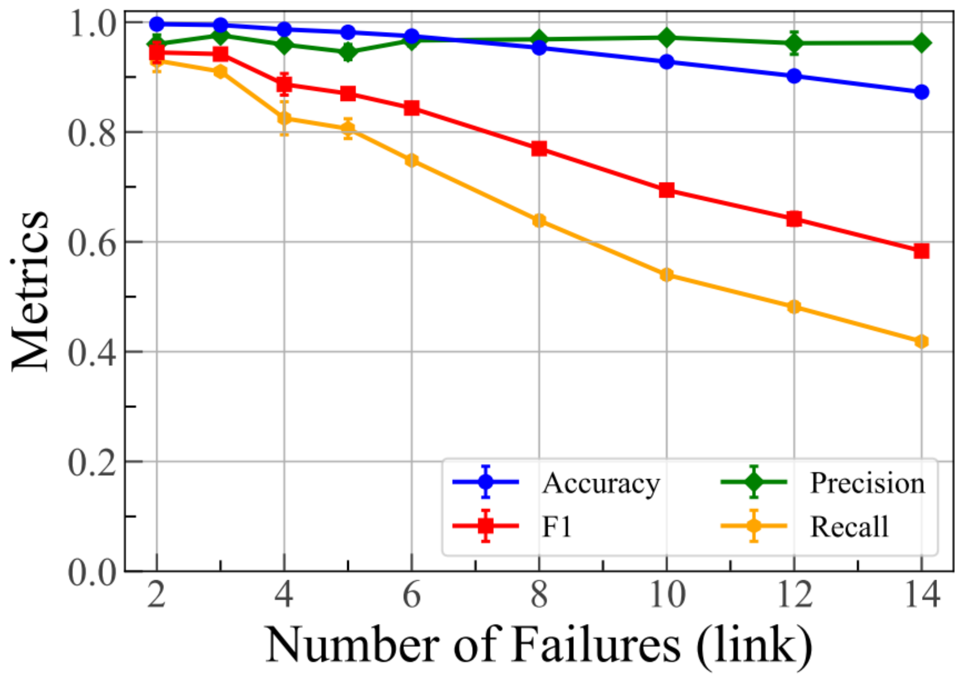
- **Multiple link failures caused by a single node failure**

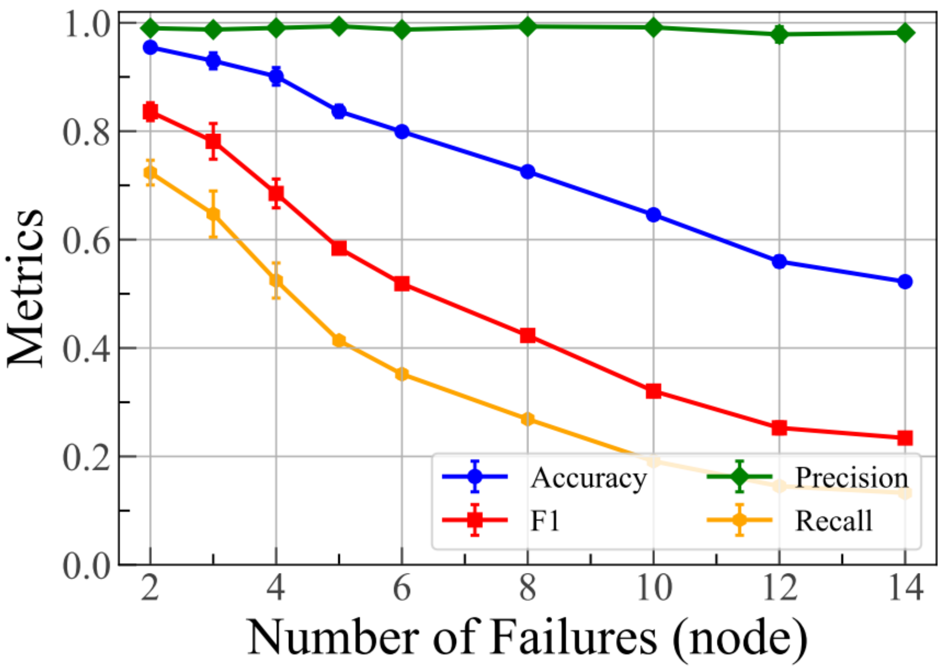




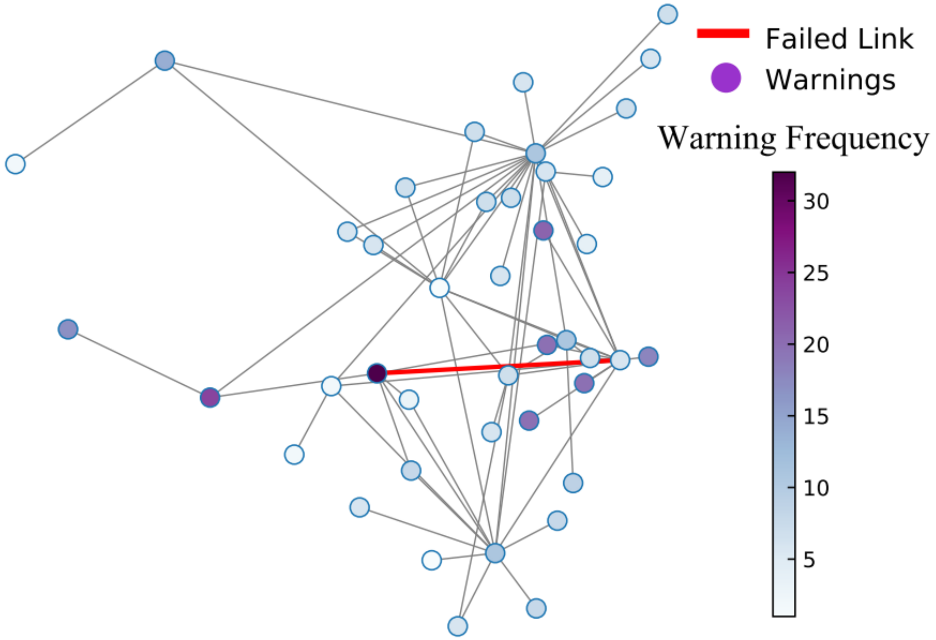
Multiple Failures Scenario (Chinanet)

- **Random multiple failures**

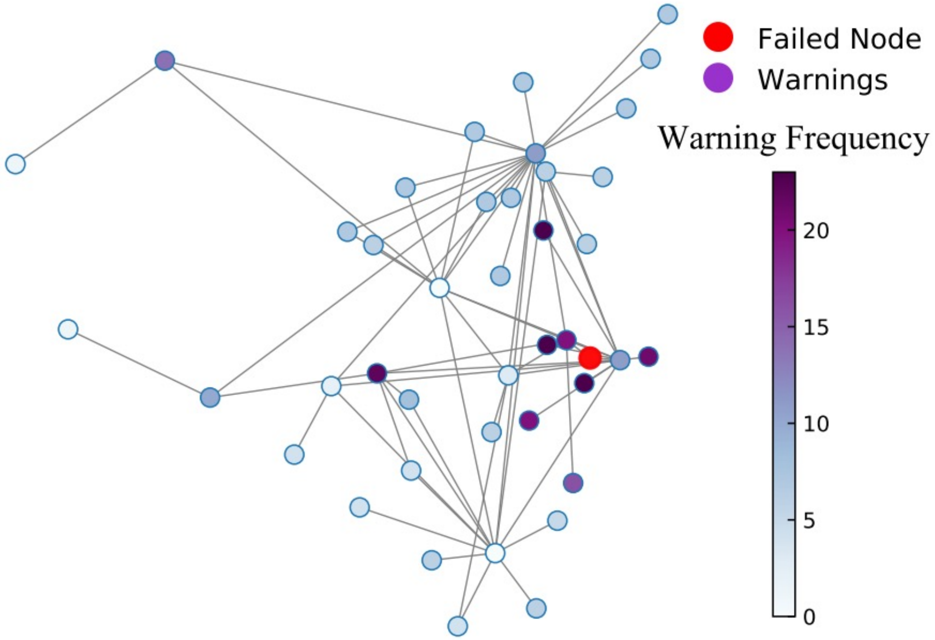




Warning Locality



Chinanet (link)



Chinanet (node)

Conclusion

- We introduce **Drift-Bottle**, a lightweight and distributed approach to failure localization in general networks
- **Drift-Bottle** utilizes the **in-network intelligence technique to detect flow-level anomalies on switches**, then generates concise inferences about potential failures with information of data paths
- Instead of a centralized mechanism, **Drift-Bottle** uses a **distributed mechanism for inferences aggregation**, which avoids high bandwidth overhead and additional infrastructural modification in networks

Thanks!