

Special Topics: Machine Learning (ML) for Networking

COL867

Holi'25

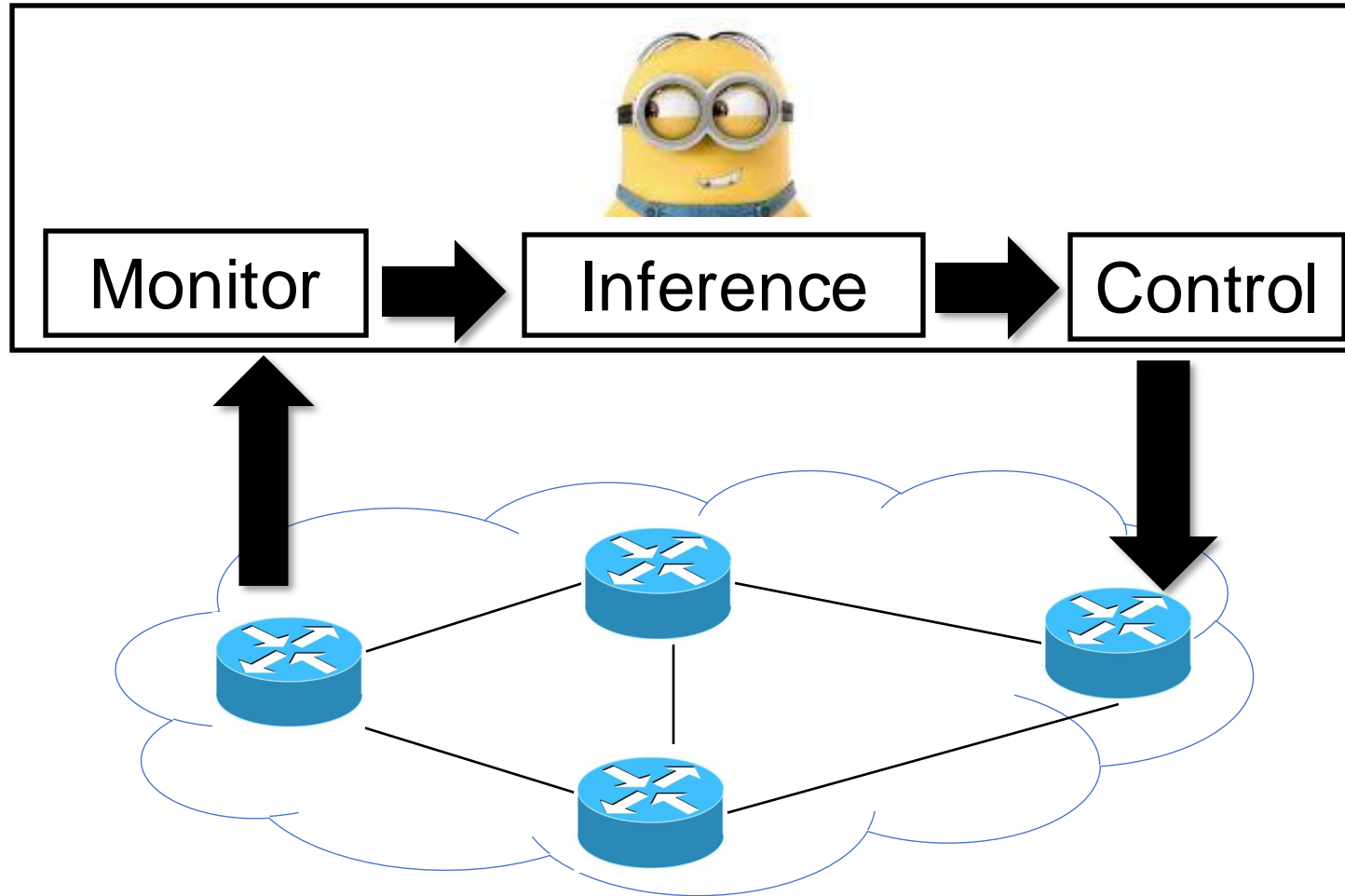
Lecture 2

Tarun Mangla

Lecture Outline

- What is ML for Networking (MLNet)
- Why MLNet Now
- Ideals of MLNet
- Intro to Networking Data

Network Management Control Loop



Learning Problems for Networks

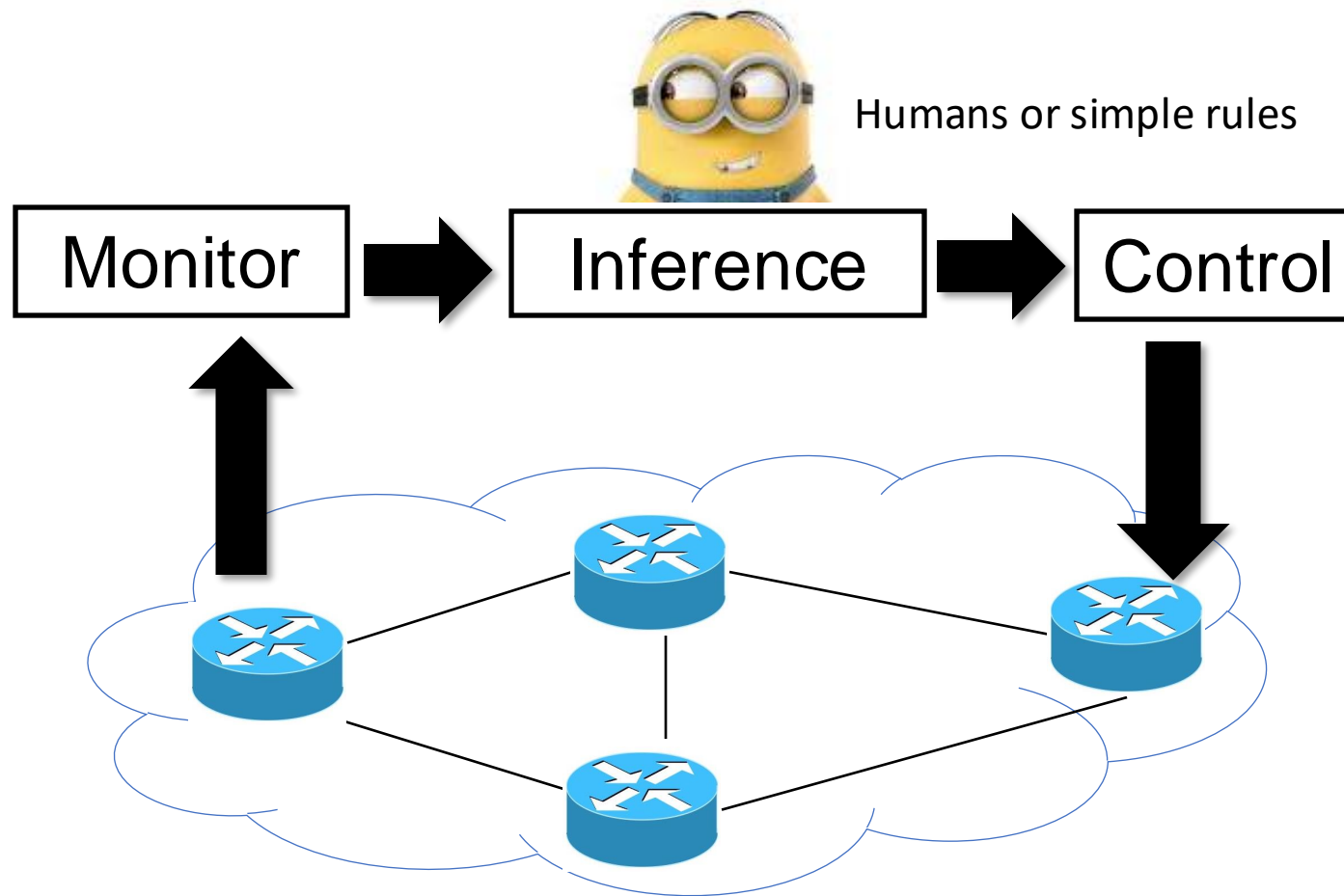
- ***Learning for networks is not new***

- Learning is intrinsic to networking
- Attributable to inherent partial visibility into network's state
- Entails multiple complex closed-loops across different spatial and temporal granularities

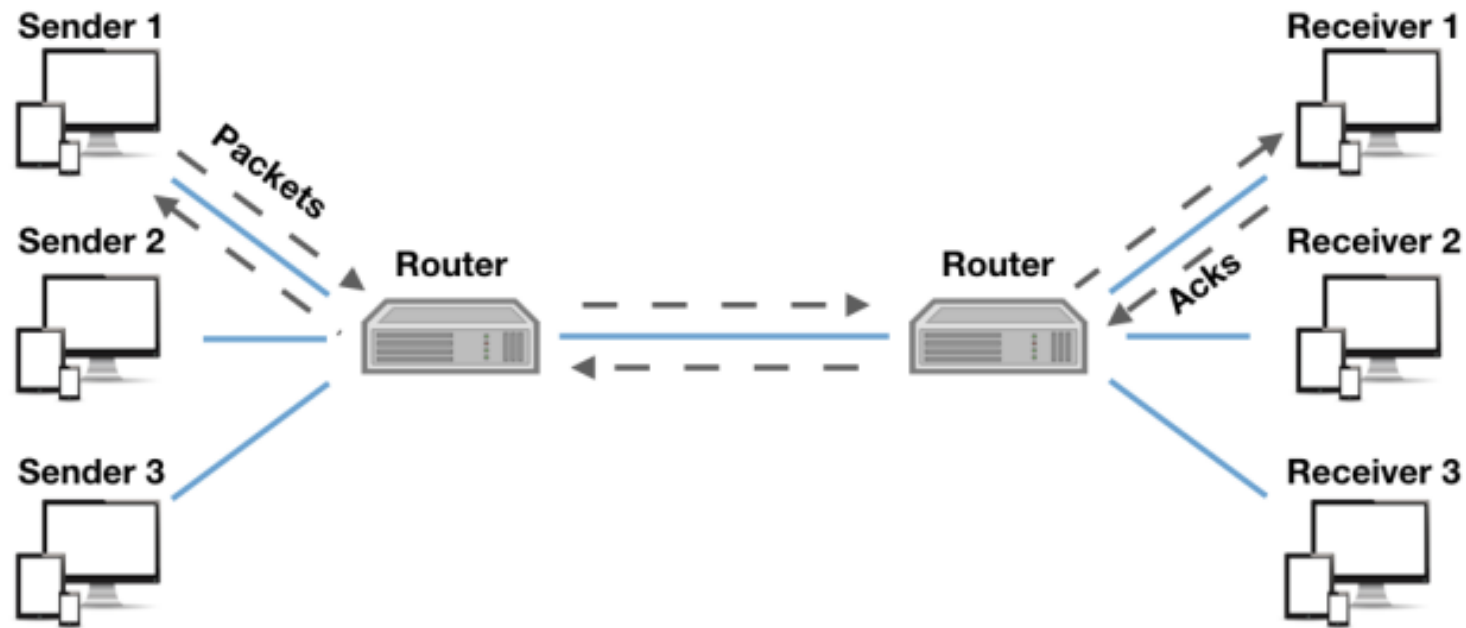
- **Examples**

- For TCP, end-hosts need to infer available bandwidth
- For video streaming, client needs to infer future network bandwidth
- For network security, the operator needs to infer benign/malicious traffic
- ...

Existing Approaches are Heuristics-based



Learning Problems for Networks (TCP Congestion Control)



Learning Problems for Networks (TCP)

- **Monitor**

- **Indicators of congestion:** Packet loss, delays

- **Infer**

- **Available bandwidth:** TCP aims to estimate the available bandwidth on the path between the sender and receiver.

- **Control**

- **Window size:** Update window size so as to achieve a balance between high utilization and low delay while ensuring fairness and stability

Approaches to Solve Learning Problems

- **Simple (transparent, explainable) heuristics**
 - E.g., current breed of congestion control and ABR algorithms employ simple rule-based approach to infer network state

Changing Requirements



Need to handle tasks that are:

~~simpler~~ (**complex**),
~~small~~(**large**)-scale,
~~infrequent~~ (**frequent**)



Approaches to Solve Learning Problems

- **Simple (transparent, explainable) heuristics**
 - E.g., current breed of congestion control and ABR algorithms employ simple rule-based approach to infer network state
- **Machine Learning for Networks**
 - Recent works have shown ML-based solutions can better detect subtle and complex patterns in data → better decisions
 - E.g., ex Machina (TCP), Pensieve (ABR), ...

Learning Problems for Networks (Security)

Scaling Intrusion Detection System (IDS) and Firewalls

- Require flexible packet processing
- Specialized h/w or user-space processing (**expensive**)



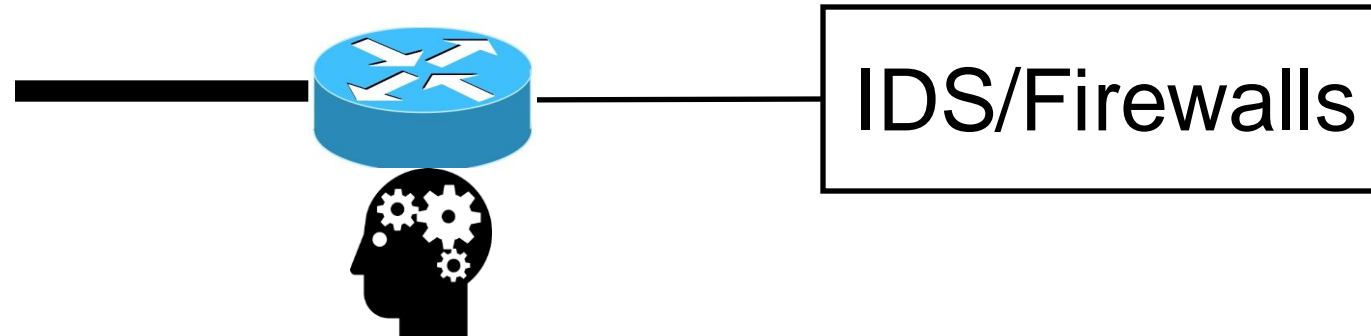
Learning Problems for Networks (Security)

- **Observation**

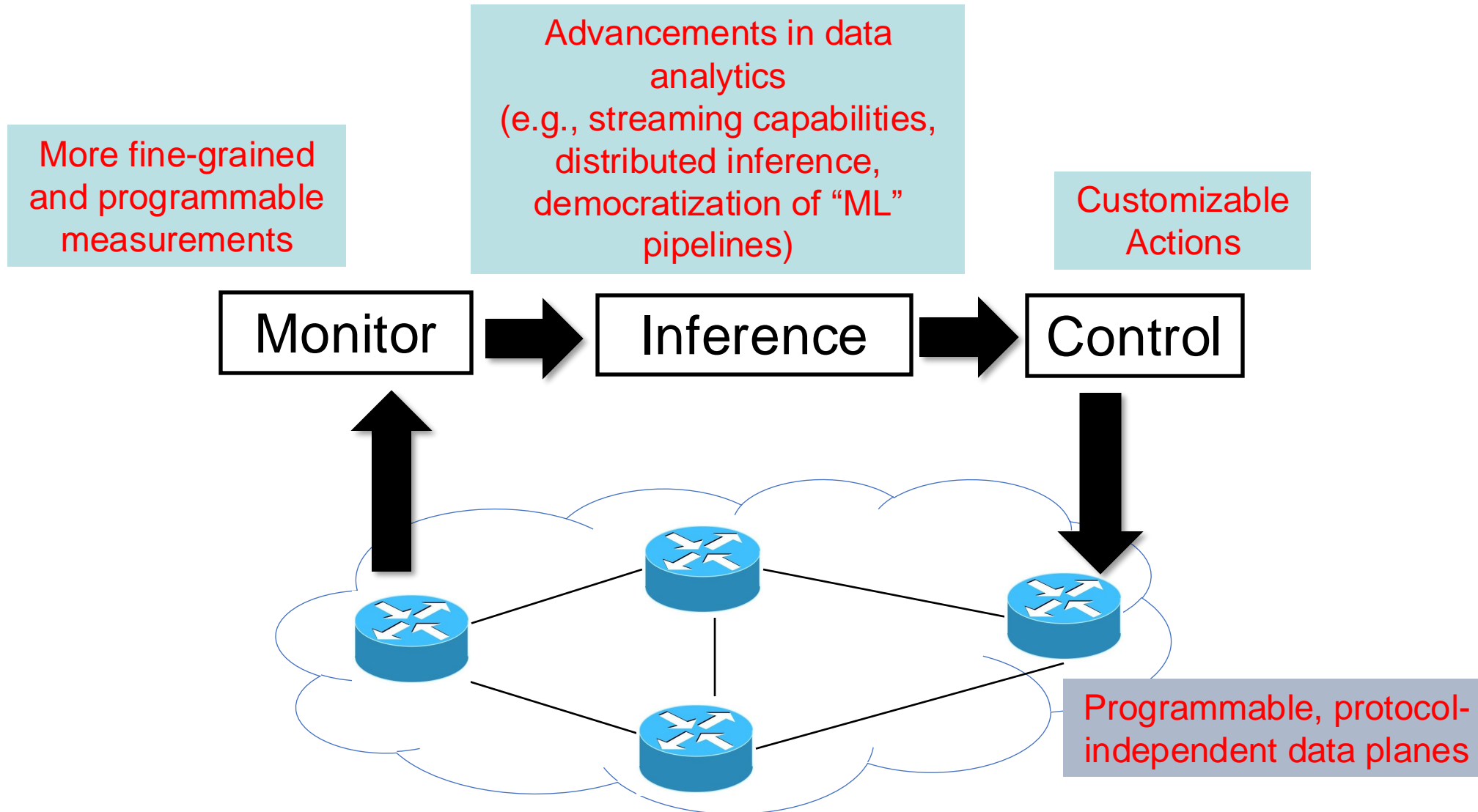
Not all traffic needs to go to the the IDS boxes

- **Approach**

Use ML model(s) to detect benign traffic



Why ML for Networking “Now”?



The State of the NetML:

Where We Are, Where We'd Like to Be

- **Monitoring**
 - low-level metrics/aggregates
 - ...but not high-level application characteristics properties (e.g., QoE)
- **Inference**
 - offline detection (security) and prediction (TE, provisioning)
 - ...but not in real time and not coupled w/control
- **Control**
 - networks are more programmable
 - ... but not always scalable, distributed, or real time
 - ... not always integrated across routing protocols, scheduling/shaping, job/task/cache placement, etc.

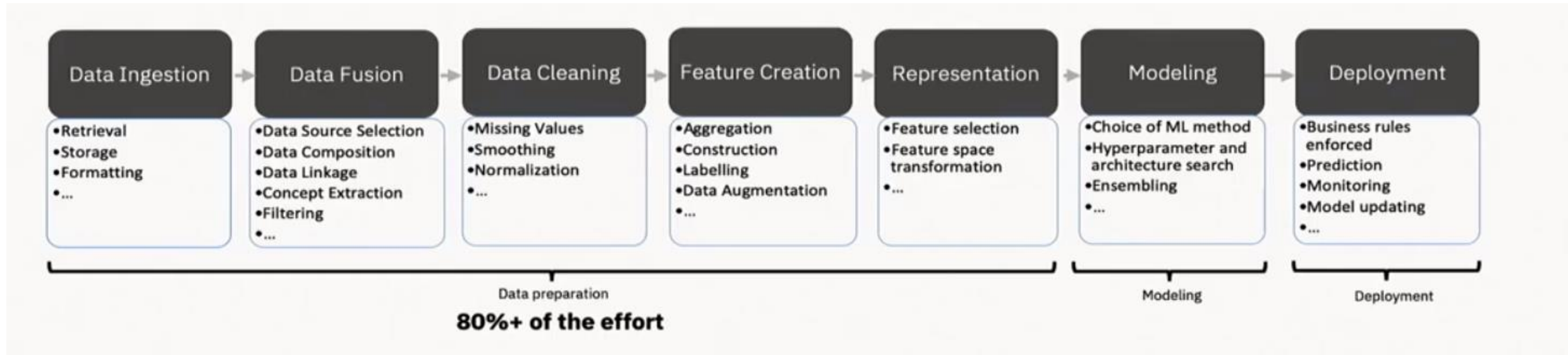
Security and Privacy Ideals

- **Monitoring:** Gather only what's needed
 - Use queries to drive data collection (IPFIX, packet traces, payload, DNS)
 - Trigger “deep dives” based on lightweight thresholds
- **Inference:** Detect attacks in real-time (not just offline traces)
 - “Compile” machine learning-based inference models to line-rate targets
 - Trigger gathering of more information as needed (networking meets active learning)
- **Control:** Automate (some) decisions
 - Rate limiting decisions (DNS response rate-limiting in-network)
 - Traffic redirection (e.g., to scrubbers, middleboxes)
 - On-the-fly placement of virtual middleboxes

Summary

- Learning fundamental to networking
 - Closed loop: Monitor → Infer → Control
- Traditional networking learning characterized by heuristics. However, sub-optimal
- Growing adoption of machine learning
 - Application pull and Technology push

Machine Learning Pipeline

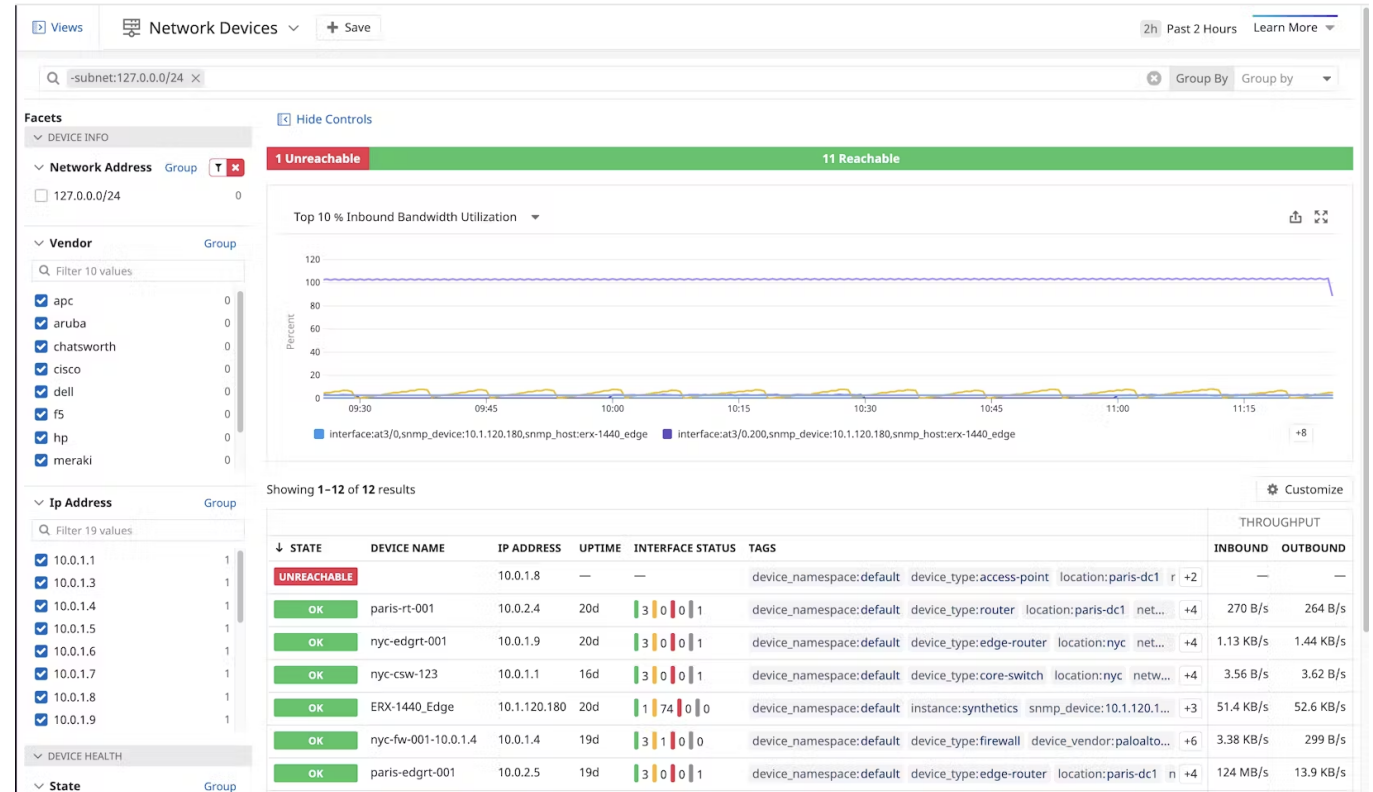


- What is the starting point?
- Let's look at the network data

Network Data

SNMP Logs

- Simple Network Management Protocol (SNMP) logs
 - CPU Usage
 - Throughput
 - Temperature
 - Uptime
 - ..



Flow-level Statistics

- Multiple protocols such as NetFlow, sFlow, IPFIX
- Sample NetFlow logs

Date flow start	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Flags	Tos	Packets	Bytes	pps	bps	Bps	Flows
2011-12-27 14:59:51.000	0.000	TCP	SOH 193.170.228.12:80	-> SOH	193.170.78.12:61414	.A...F	0	1	54	0	0	54	1
2011-12-27 14:58:33.000	59.000	TCP	SOH 193.170.78.51:32812	-> SOH	193.170.78.51:32812	.AP...	0	6	2236	0	303	372	1
2011-12-27 14:59:44.000	5.000	TCP	SOH 193.170.78.2:64067	-> SOH	193.170.78.2:64067	.AP...	164	13	13144	2	21030	1011	1
2011-12-27 14:59:52.000	0.000	TCP	SOH 193.170.78.2:53103	-> SOH	193.170.78.2:53103	.AP...	0	1	200	0	0	200	1
2011-12-27 14:59:57.000	0.000	TCP	SOH 193.170.78.51:46756	-> SOH	193.170.78.51:46756	.AP...	0	1	522	0	0	522	1
2011-12-27 14:56:04.000	208.000	TCP	SOH 193.170.78.12:61787	-> SOH	193.170.78.12:61787	.A....	164	475	719150	2	27659	1514	1
2011-12-27 14:59:54.000	0.000	TCP	SOH 193.170.78.2:58525	-> SOH	193.170.78.2:58525	.A....	0	1	1514	0	0	1514	1
2011-12-27 14:59:32.000	0.000	UDP	SOH 193.170.78.3:53	-> SOH	193.170.78.3:53	0	1	89	0	0	89	1

Packet-level data

- Most detailed data
- Challenging to collect and store at scale

The image shows a Wireshark packet capture interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons. The main window is divided into three panes. The top pane shows a list of captured packets with columns for No., Time, Source, Destination, Protocol, Length, and Info. The bottom pane shows the details of the selected packet (No. 5), including Ethernet II, Internet Protocol Version 4, and User Datagram Protocol (UDP) fields. The bottom pane also shows the raw packet data in hexadecimal and ASCII format.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
2	0.006060	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
3	0.011873	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
4	0.017851	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
5	0.023572	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
6	0.029199	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
7	0.035617	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
8	0.041516	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
9	0.047367	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
10	0.053186	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
11	0.058795	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
12	0.064700	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
13	0.070517	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
14	0.076211	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
15	0.081982	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
16	0.087916	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
17	0.090059	172.25.0.1	10.10.1.90	UDP	380	54456 → 2059 Len=338
18	3.219995	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
19	3.225918	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358
20	3.231851	172.25.0.1	10.10.1.90	UDP	1400	54456 → 2059 Len=1358

> Frame 5: 1400 bytes on wire (11200 bits), 1400 bytes captured (11200 bits) on interface 0
> Ethernet II, Src: Cisco_7e:71:42 (00:07:7d:7e:71:42), Dst: Vmware_84:64:f7 (00:50:56:84:64:f7)
> Internet Protocol Version 4, Src: 172.25.0.1, Dst: 10.10.1.90
▼ User Datagram Protocol, Src Port: 54456, Dst Port: 2059
Source Port: 54456
Destination Port: 2059
Length: 1366
[Checksum: [missing]]
[Checksum Status: Not present]
[Stream index: 0]

0000 00 50 56 84 64 f7 00 07 7d 7e 71 42 00 00 45 00 .PV.d...}~qB..E.
0010 05 6a 17 40 00 00 fd 11 e9 bc ac 19 00 01 0a 0a .j.H.....
0020 01 5a d4 b8 08 00 05 56 00 00 00 00 2c 77 57 .Z.....V.....W.
0030 17 b8 5a 30 79 8c 00 00 b1 65 01 00 00 01 00 00 ..Z0y...e.....
0040 00 2c 01 00 00 09 00 08 00 04 00 0c 00 04 00 07
0050 00 02 00 0b 00 02 00 05 00 01 00 04 00 01 00 0a
0060 00 04 00 02 00 08 00 0e 00 04 01 00 05 0e 34 144.
0070 4b 97 0a 0b 01 20 01 bb e9 35 00 06 00 00 00 0a K....5.....
0080 00 00 00 00 00 00 01 00 00 00 0a 14 26 2b&+
0090 0a 0a 05 05 01 bb c1 21 00 06 00 00 00 0a 00 00!
00a0 00 00 00 00 00 00 00 00 ac 1f 0a 25 0a 0a%.
00b0 30 07 1f 90 ed 61 00 06 00 00 0a 00 00 00 00 00a.....
00c0 00 00 00 0c 00 00 00 00 ac 1f 0a 25 0a 0a 05 45%.E
00d0 1f 90 dc d4 00 00 00 00 0a 00 00 00 00 00 00 00
00e0 00 01 00 00 00 00 ac 1f 0a 25 0a 0a 30 14 1f 90%.0..
00f0 c4 1a 00 06 00 00 0a 00 00 00 00 00 00 00 01
0100 00 00 00 00 0a 14 24 73 0a 0a b0 0c d2 3b 01 b0\$s.....j..

Exploring Packet-level Data

- Clone on your laptop: <https://github.com/tarunmangla/col867-holi25/tree/master>
- Open `network-data-exploration.ipynb` in jupyter-lab
- **Exercise:** Find the speed from the network trace