Towards Self-Driving Networks:
AI-enabled network management

Abdelkader LAHMADI
abdelkader.lahmadi@loria.fr

NMRG meeting, 01/07/2019

Project-Team RESIST
Inria Nancy Grand Est

Self-Driving Technology: a reality

In Movies:

Total Recall (1990)

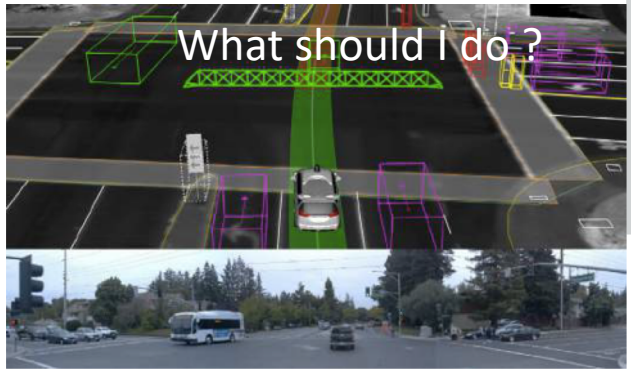
In Real World:

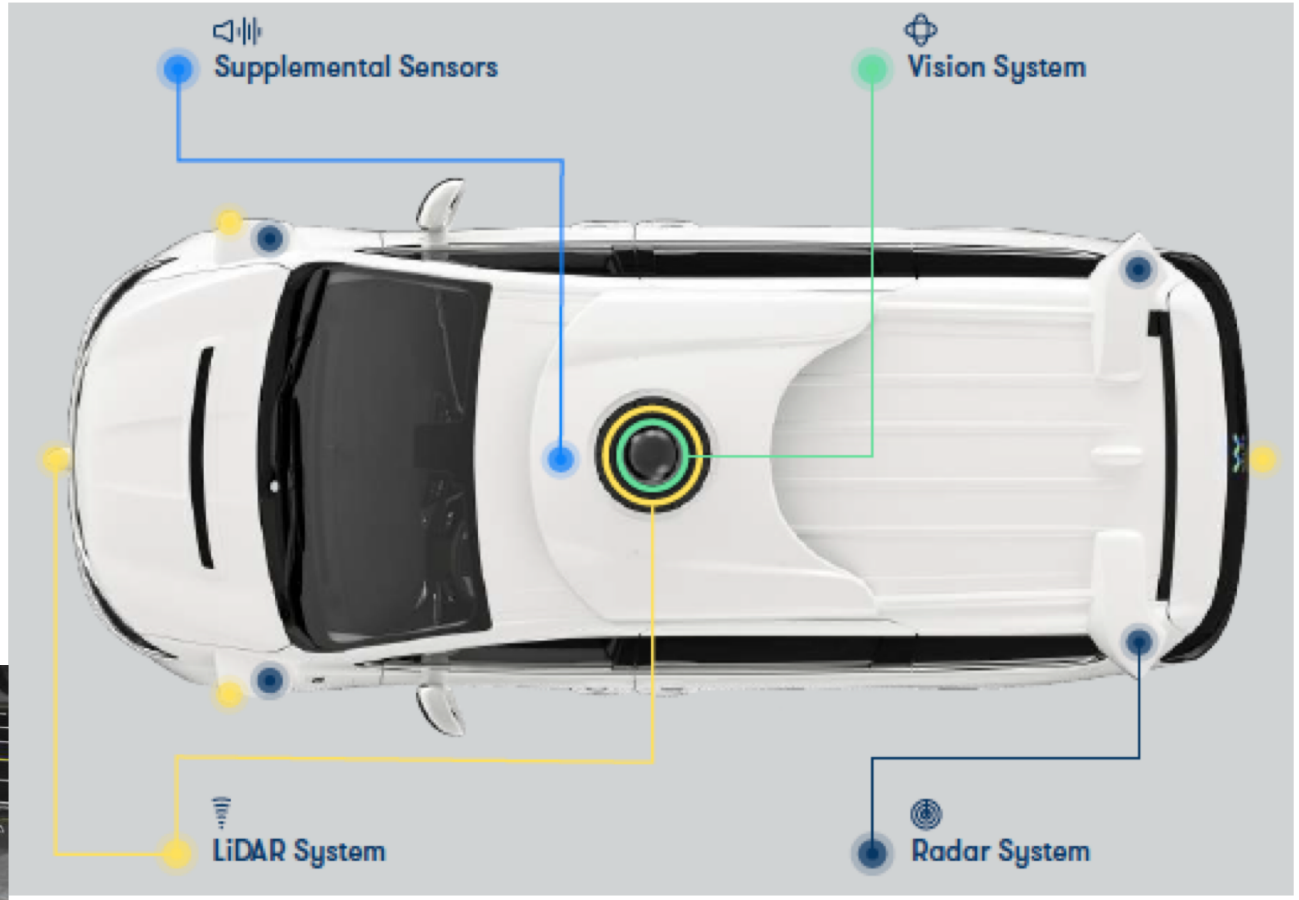
Google car (2014)

How self-driving car work ?

- Perception
  - Detect and classify objects on the road
  - Estimate speed, heading, acceleration over time
- Behaviour prediction
  - Predict and understand the intent of each object in the road
- Planner
  - Make decisions: turn left, right, slow down, etc.

How self-driving car work ?





Waymo company, "Waymo Safety report: on the road to fully Self-Driving", 2018

How to build a self-driving car ?

- Get a safe driving model from human drivers
- Integrate a lot of sensors, computing and networking capabilities in the car
- Apply ML and AI techniques
- Train and test until the car drives safely
  - Waymo cars : 4 million miles of driving, 2.5 billion simulated miles

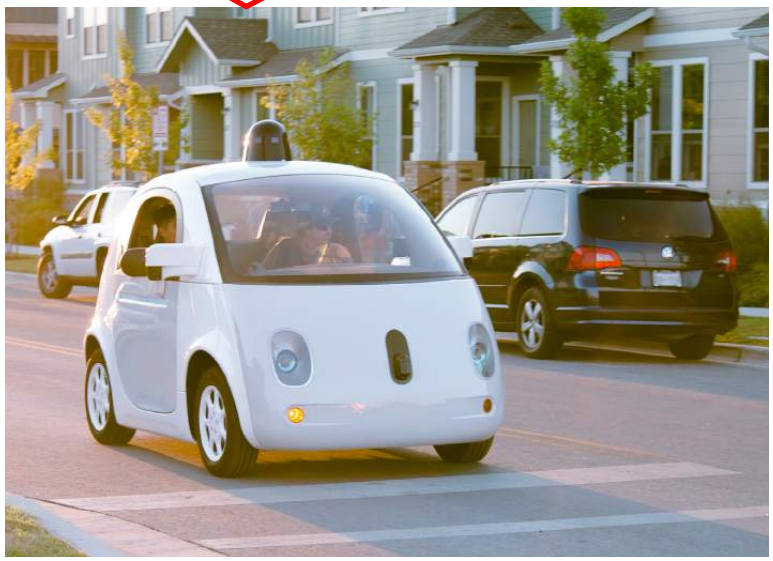
Vinton G. Cerf, « A Comprehensive Self-Driving Car Test», Communications of the ACM, February 2018

How to build a safe self-driving car?

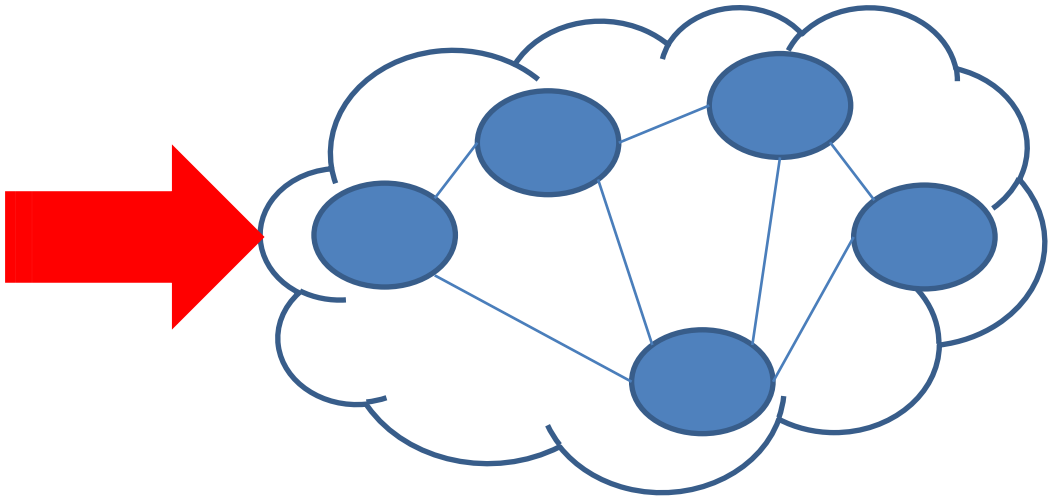
- Build verifiable software and systems
- Encrypt and verify channels of communications
- Build redundant security measures for critical systems
- Limit communication between critical system
- Provide timely software updates
- Model and prioritize threats

From cars to networks

4 wheels, gears, motors, and more



Switches, routers, links, and devices



- Real-time monitoring
- Softwarisation
- Automation
- Personalised services

How to build self-driving networks ?

Difficult to have, oftenly does not exist
Lot of human effort to maintain performance and security

- Get a safe model of a modern network from human operators
- Integrate a lot of probes, and management capabilities
- Apply ML and AI techniques
- Train and test until it works safely

Self-Driving Networks

Why (and How) Networks Should Run Themselves

Nick Feamster and Jennifer Rexford
Princeton University

Abstract

The proliferation of networked devices, systems, and applications that we depend on every day makes managing networks more important than ever. The increasing security, availability, and performance demands of these applications suggest that these increasingly difficult network management problems be solved in real time, across a complex web of interacting protocols and systems. Alas, just as the importance of network management has increased, the network has grown so complex that it is seemingly unmanageable. In this new era, network management requires a fundamentally new approach. Instead of optimizations based on closed-form analysis of individual protocols, network operators need data-driven, machine-learning-based models of end-to-end and application performance based on high-level policy goals and a holistic view of the underlying components. Instead of anomaly detection algorithms that operate on offline analysis of network traces, operators need classification and detection algorithms that can make real-time, closed-loop decisions. Networks should learn to drive themselves. This paper explores this concept, discussing how we might attain this ambitious goal by more closely coupling measurement with real-time control and by relying on learning for inference and prediction about a networked application or system, as opposed to closed-form analysis of individual protocols.

1 Introduction

Modern networked applications operate at a scale and scope we have never seen before. Virtual and augmented reality require real-time responsiveness, micro-services deployed using containers introduce rapid changes in traffic workloads, and the Internet of Things (IoT) significantly increases the number of connected devices while also raising new security and privacy concerns. The widespread integration of these applications into our daily lives raises the bar for network management, as users elevate their expectations for real-time interaction, high availability, resilience to attack, ubiquitous access, and scale. Network management has always been a worthwhile endeavor, but now it is mission critical.

Yet, network management has remained a Sisyphean task. Network operators develop and use scripts and tools to help them plan, troubleshoot, and secure their networks, as user demands and network complexity continue to grow. Networking researchers strive to improve the tuning, design, and measurement of network protocols, yet we continue to fall behind the curve, as the protocols, variable network conditions, and relationships between them and user quality of experience become increasingly complex. Twenty years ago, we had some hope of (and success in) creating clean, closed-form models of individual protocols, applications, and systems [4, 24]; today, many of these are too complicated for closed-form analysis. Prediction problems such as determining how search query response time would vary in response to the placement of a cache are much more suited to statistical inference and machine learning based on measurement data [29].

Of course, we must change the network to make network management easier. We have been saying this for years, as we continue to fall behind the curve. Part of the problem, we believe, is the continued focus on designing, understanding, and tweaking individual protocols—we focus on better models for BGP, optimizations for TCP, QUIC, DNS, or the protocols du jour. In fact, our troubles do not lie in the protocols. The inability to model holistic network systems, as opposed to individual protocols, has made it difficult for operators to understand what is happening in the network. Software-Defined Networking (SDN) helps by offering greater programmability and centralized control, yet controller applications still rely on collecting their own data and installing low-level match-action rules in switches and SDN does not change the fact that real networked systems are too complex to analyze with closed-form models.

As networking researchers, we must change our approach to these problems. An ambitious goal for network management is that of a *self-driving network*—one where (1) network measurement is task-driven and tightly integrated with the control of the network; and (2) network control relies on learning and large-scale data analytics of the entire networked system, as opposed to closed-form models of individual protocols. Recent initiatives have proffered this high-level goal [14, 28], drawing an analogy to self-driving cars, which can make decisions that manage uncertainty and mitigate risk to achieve some task (*e.g.*, transportation to some destination). This paper explores this goal in detail, developing the technical requirements for and properties of a self-driving network and outlining a broad, cross-disciplinary research agenda for the community that can move us closer to realizing this goal.

The networking research community has been developing the pieces of this puzzle for many years, from predictive models of application performance [19, 29] to statistical anomaly and intrusion detection algorithms based on analysis of network traffic [2, 7]. The state of the art, however, merely lays the foundation for the much more ambitious agenda of

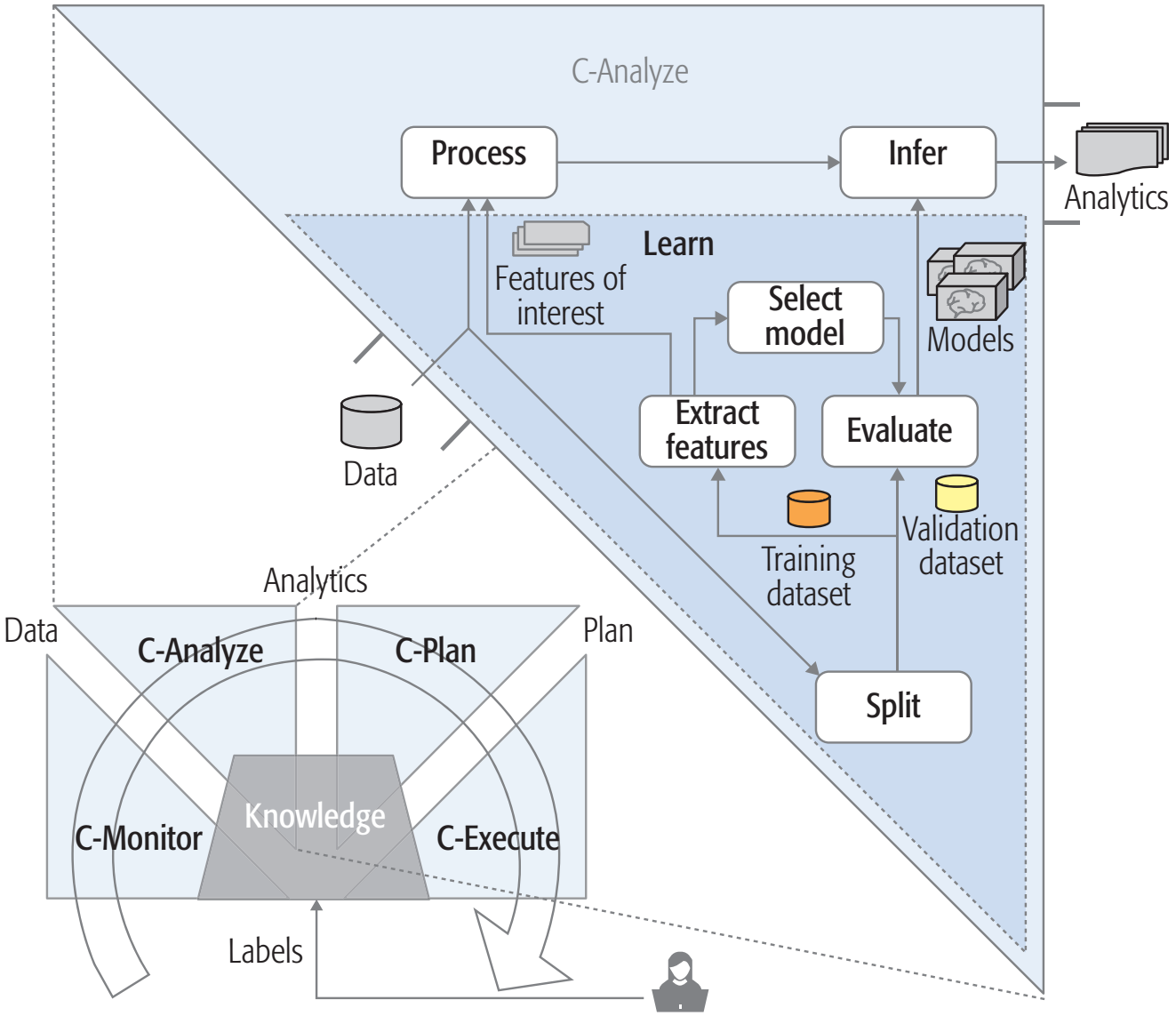
Abstract

The proliferation of networked devices, systems, and applications that we depend on every day makes managing networks more important than ever. The increasing security, availability, and performance demands of these applications suggest that these increasingly difficult network management problems be solved in real time, across a complex web of interacting protocols and systems. Alas, just as the importance of network management has increased, the network has grown so complex that it is seemingly unmanageable. In this new era, network management requires a fundamentally new approach. Instead of optimizations based on closed-form analysis of individual protocols, network operators need data-driven, machine-learning-based models of end-to-end and application performance based on high-level policy goals and a holistic view of the underlying components. Instead of anomaly detection algorithms that operate on offline analysis of network traces, operators need classification and detection algorithms that can make real-time, closed-loop decisions. Networks should learn to drive themselves. This paper explores this concept, discussing how we might attain this ambitious goal by more closely coupling measurement with real-time control and by relying on learning for inference and prediction about a networked application or system, as opposed to closed-form analysis of individual protocols.

A more pragmatic approach

- Deriving measurement, inference, and control from high-level policy
  - High-level goal (performance, security) => (measurement, inference, decisions)
- Performing automated, real-time inference
  - Improve network management through learning
  - Quality of data
- Operating scalably in the data plane: Need for Speed
  - Fully programmable protocol-independent data plane: dedicated hardware platforms, programming languages (P4)
  - In-band measurement: distributed streaming analytics

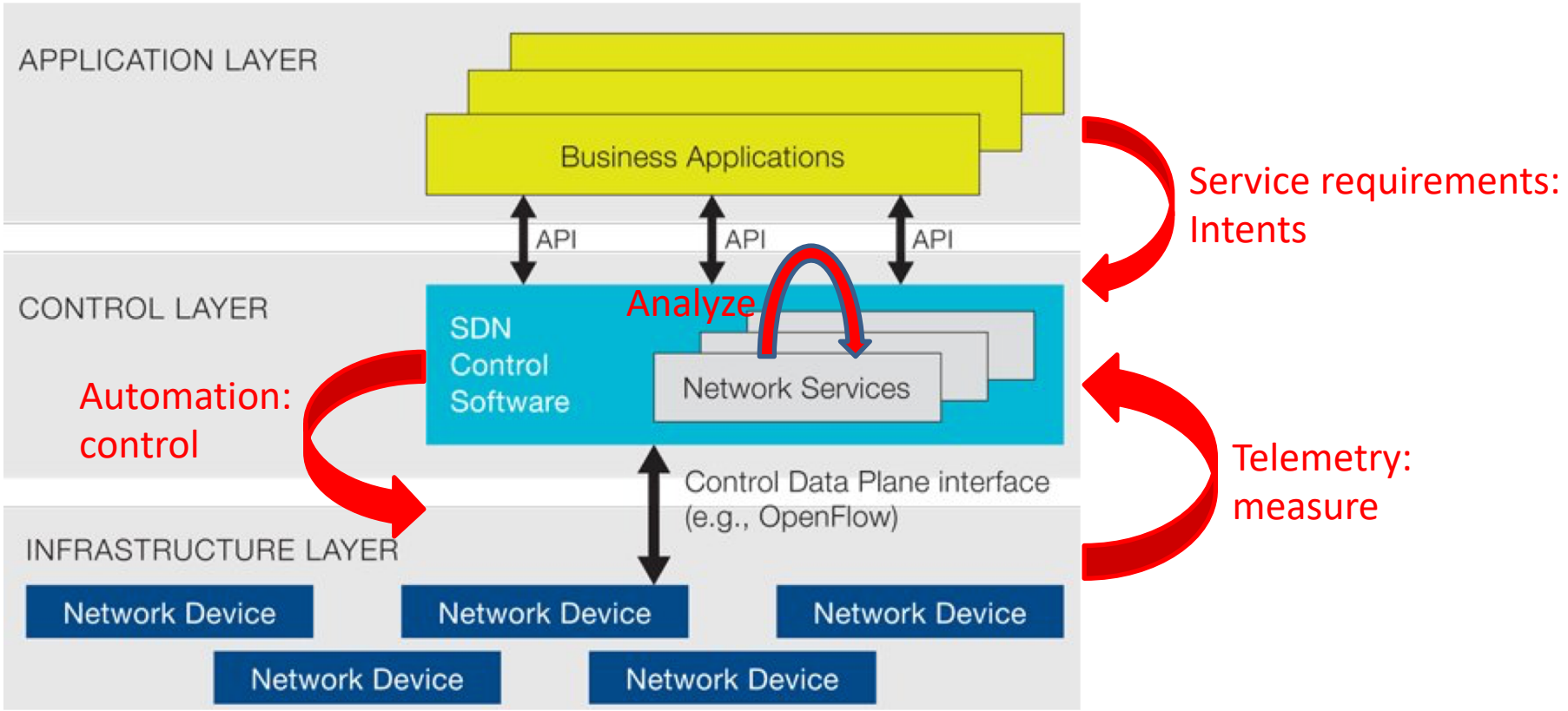
Self-driving networks: the process



From SDN to SDN

Software Defined Networks

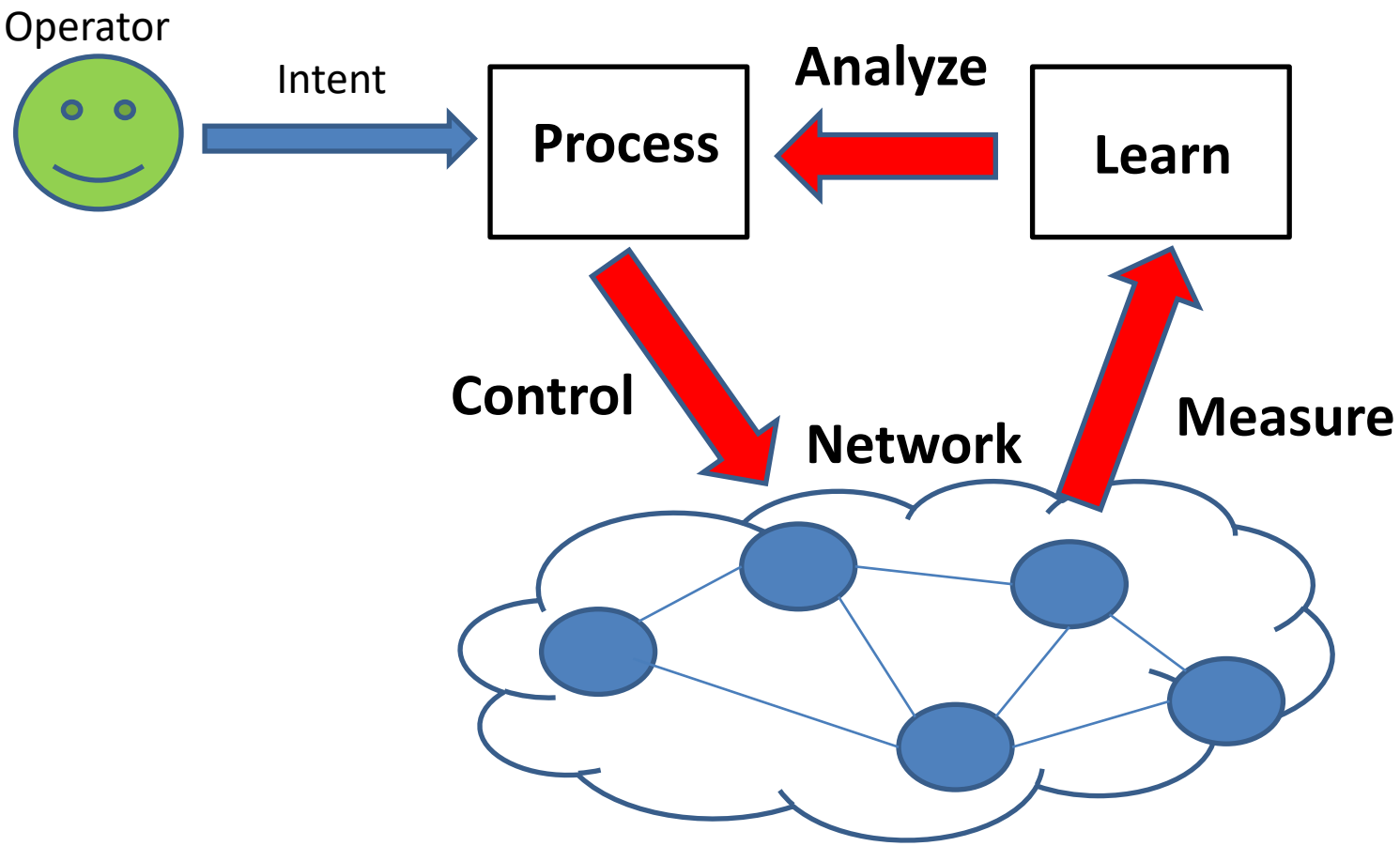
Self-Driving Networks



SDN reference architecture: www.opennetworking.org

Refining network intents

- Network intents: high level policy goals

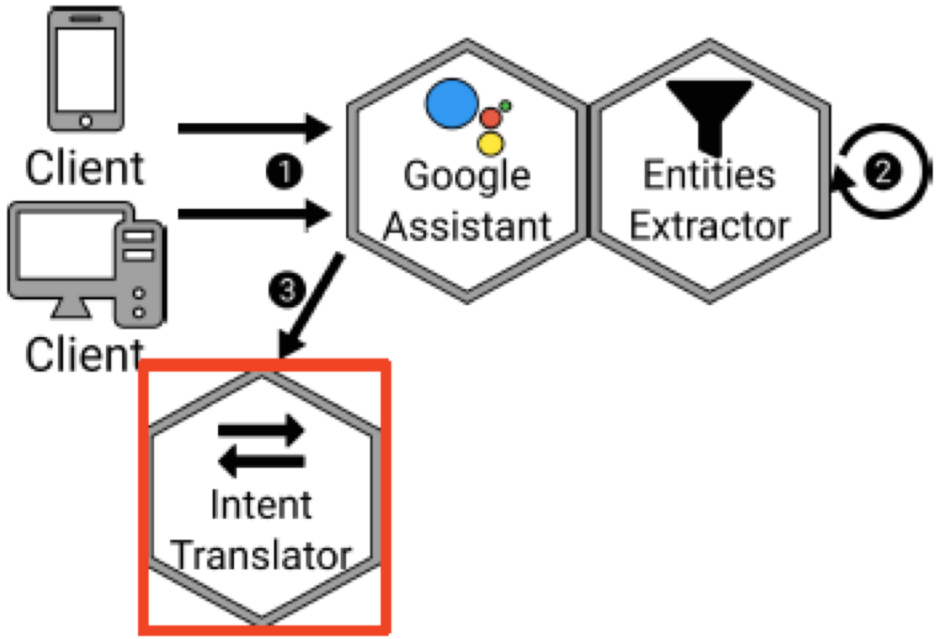


Refining network intents

Say what you want

"Please add a firewall and an IDS from Iperf client to server"

Extracted entities



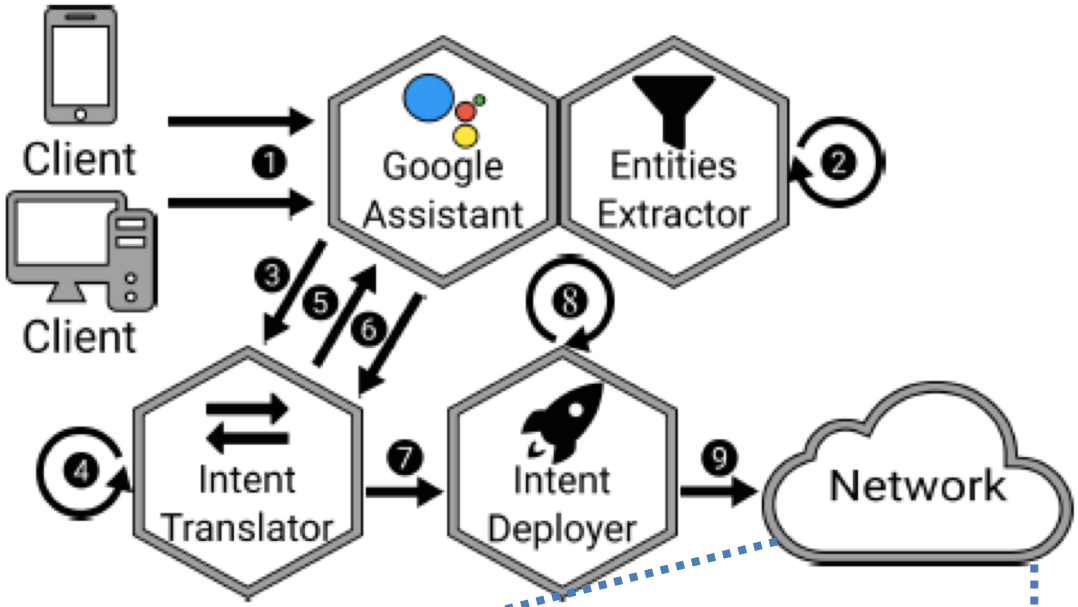
Neural Sequence to Sequence learning model, using Recursive Neural Networks.

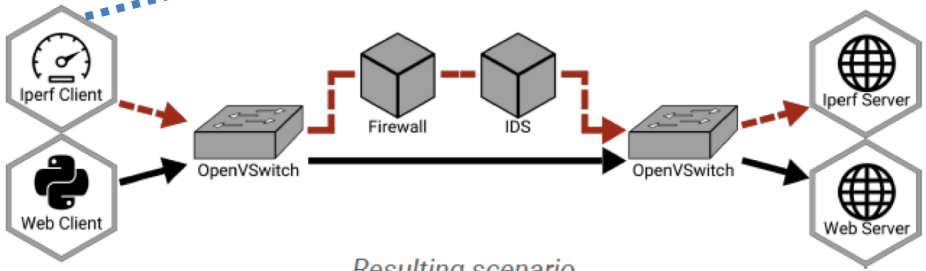
Refining network intents

```
# deploy vnfs
vim-emu compute start -n fw <params>
vim-emu compute start -n ids <params>

# chain vnfs
vim-emu network add -b -src
iperf-c:c-eth0 -dst fw:in
vim-emu network add -b -src fw:out -dst
ids:in
vim-emu network add -b -src ids:out -dst
iperf-s:s-eth0
```

Compiled SONATA-NFV commands

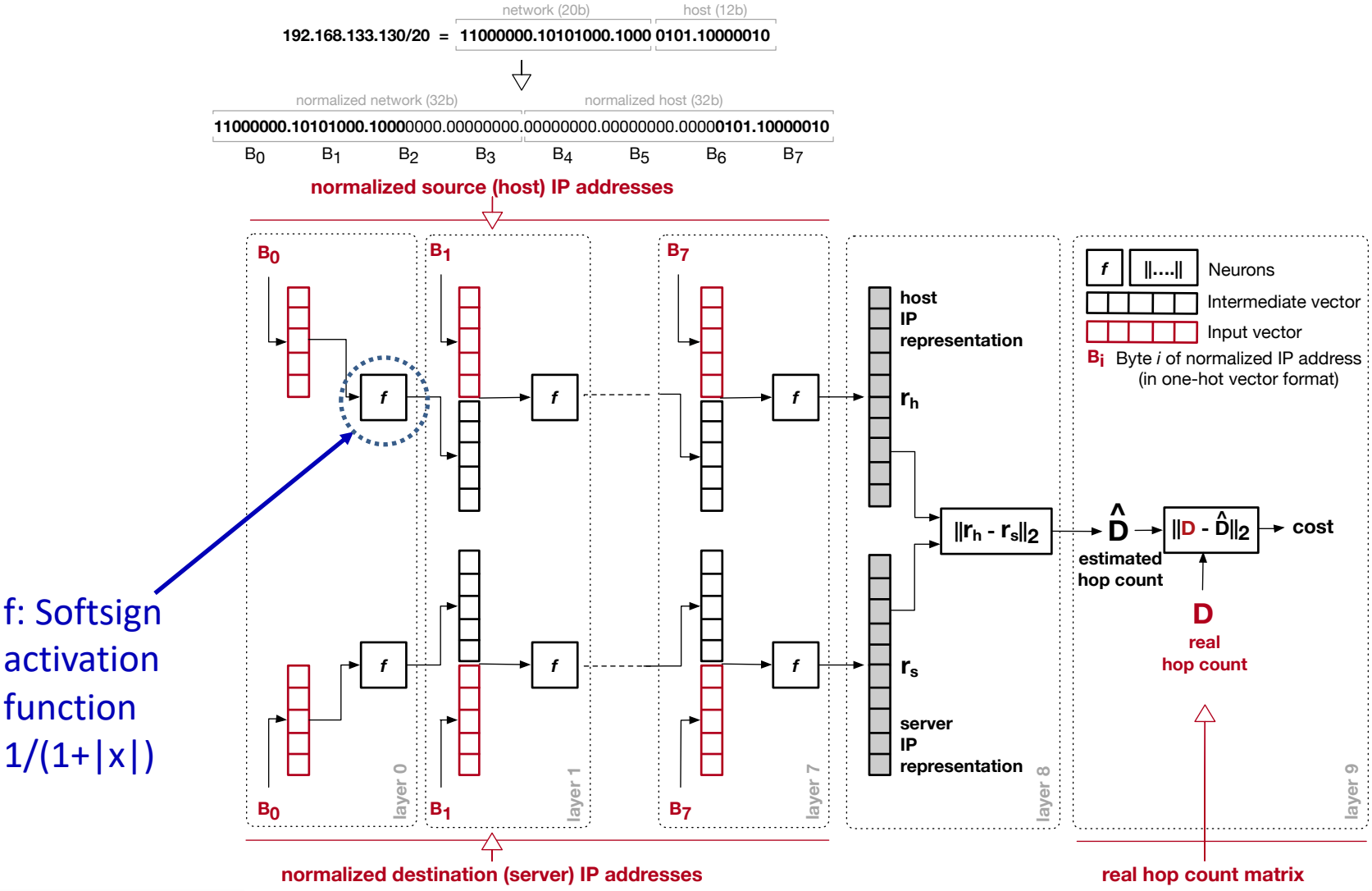




Resulting scenario

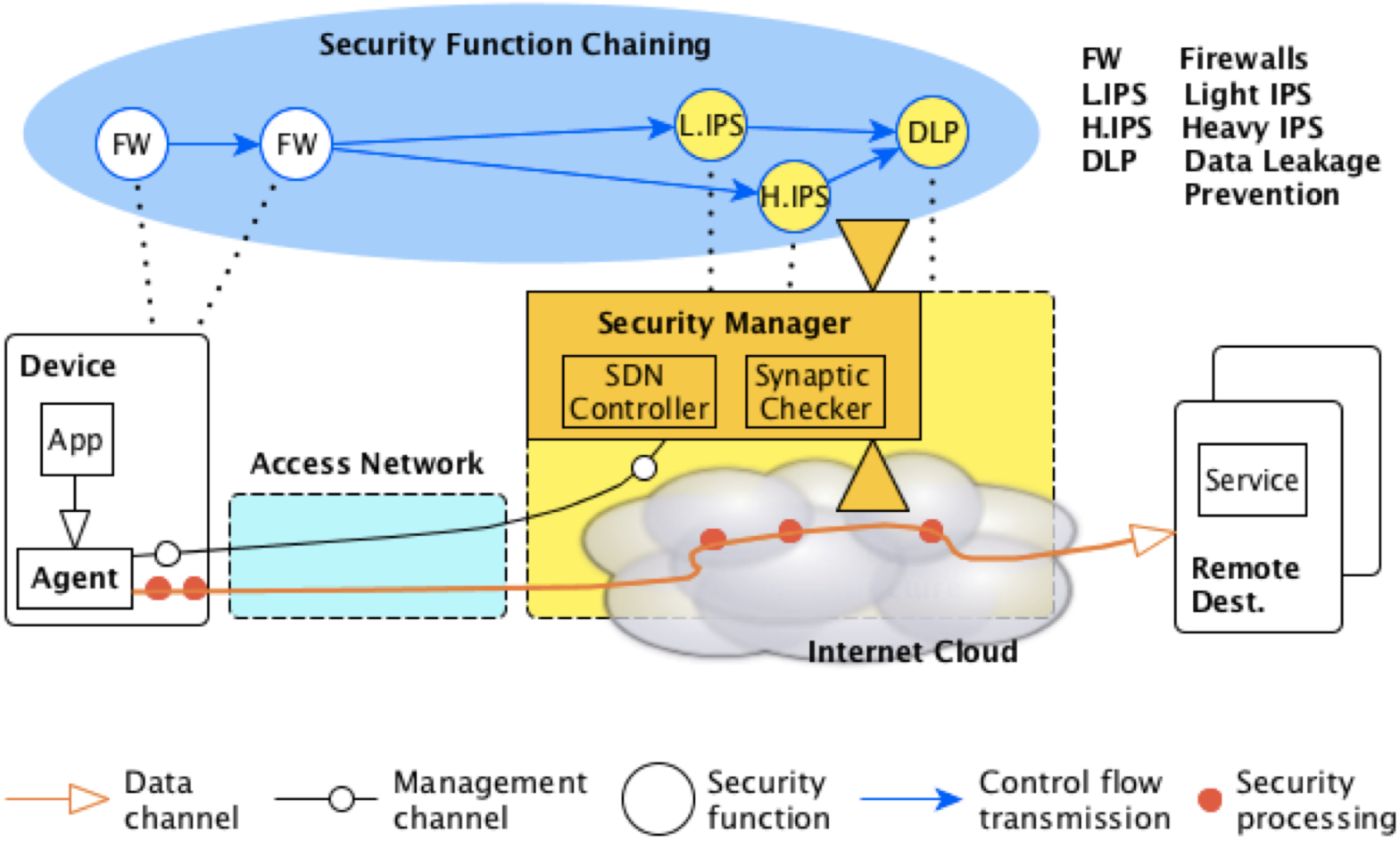
Learning IP network embedding

- Estimate network structure



f: Softsign activation function 1/(1+|x|)

Automated generation of verifiable security function chains



Automated generation of verifiable security function chains

- Inference using logic programming

Rules for inferring elementary actions

$$deploy_{block}(a, pt) \leftarrow botnet(a, pt)$$
$$deploy_{forward}(a) \leftarrow \neg worm(a, pt) \wedge \neg botnet(a, pt)$$

Rules for inferring security functions to deploy

$$stateless\_firewall(t) = \bigcirc_{+}\{\, forward(a, t) : deploy_{forward}(a),\ a \in \text{Addr} \,\} \circ_{+} \bigcirc_{+}\{\, block(a, pt, t) : deploy_{block}(a, pt),\ a \in \text{Addr},\ pt \in \text{Port} \,\}$$

Rules for inferring compositions paths

$$dos\_chain = stateless\_firewall \circ_{\gg} ids \circ_{\gg} stateful\_firewall$$

Example

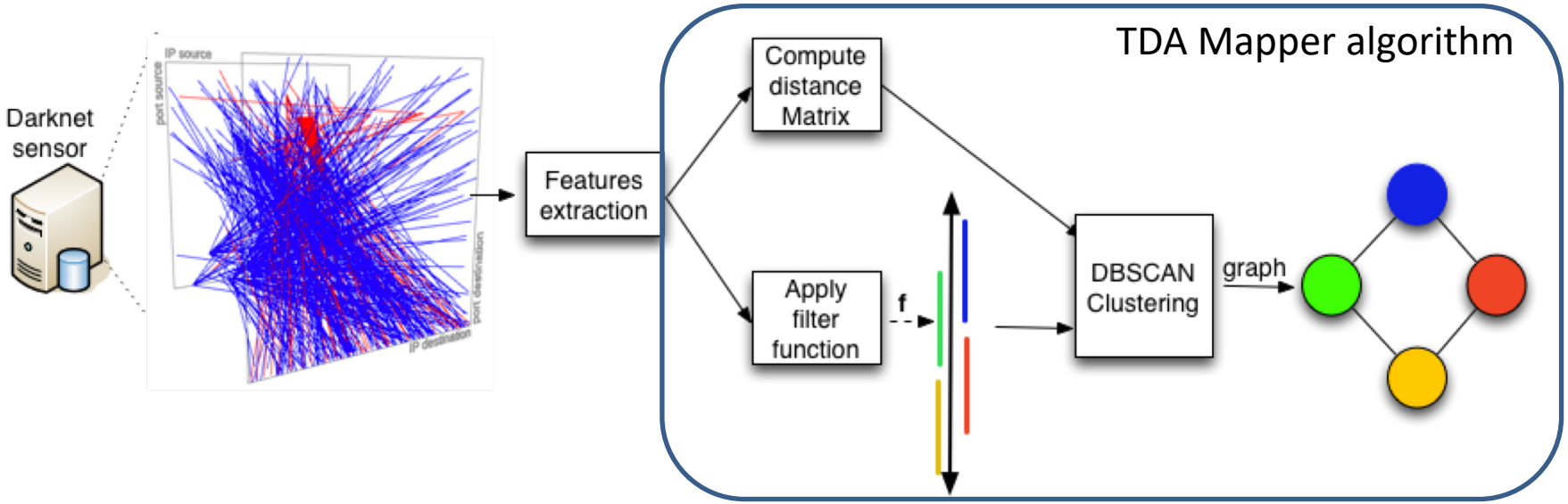
Logical statement

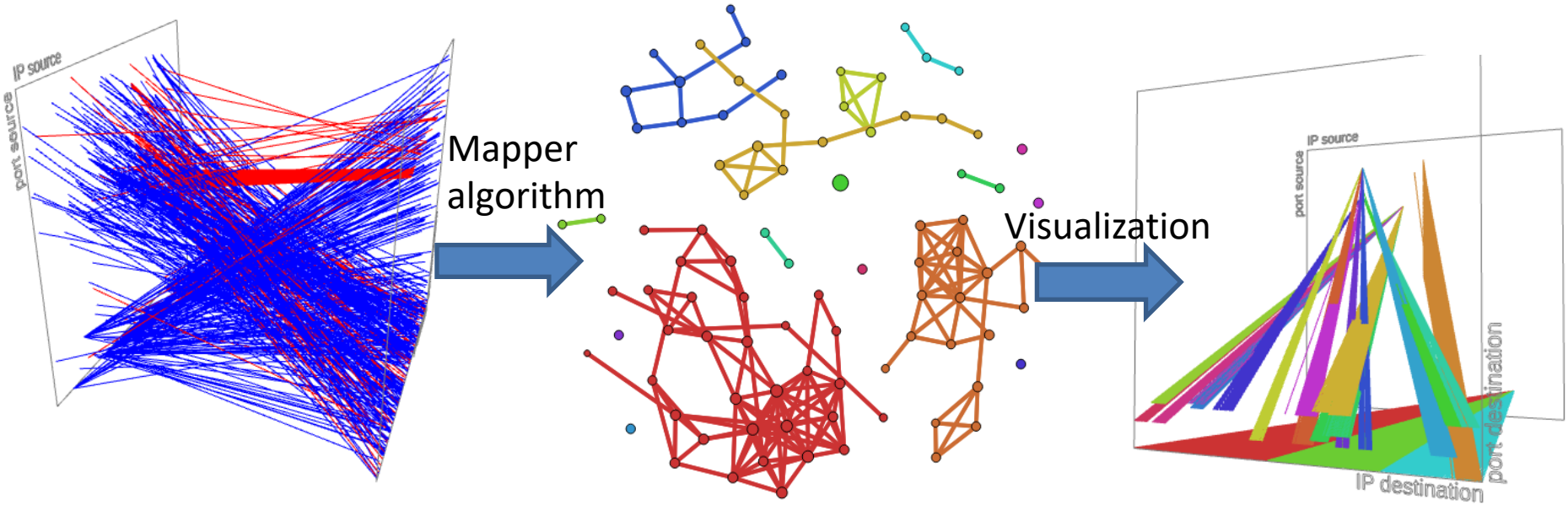
$$stateless\_firewall(t) = forward(169.45.223.16, t) \circ_{+} forward(169.45.223.20, t) \circ_{+} \ldots$$
$$dpi(t) = inspect(169.45.223.16, t) \circ_{+} inspect(169.45.223.20, t) \circ_{+} \ldots$$
$$chain(t) = [stateless\_firewall(t), dpi(t)]$$

Corresponding Pyretic program

$$stateless\_firewall = match(dstip = 169.45.223.16) + match(dstip = 169.45.223.20) \ldots$$
$$dpi = InspectQuery(169.45.223.16) + InspectQuery(169.45.223.20) + \ldots$$
$$chain = stateless\_firewall \gg dpi$$

Extracting attack patterns





Summary and outlook

- Self-driving networks: challenges
  - You can't build a self-driving Citroën 2 CV
- Enabled by network programmability and virtualisation: SDN, NVF, P4, …
- Enabled by ML and AI based networking: data-driven models
- How to place the human in the loop of self-driving networks?
  - Accountability
- How to test self-driving networks?
  - 2.5 billion simulated miles during the course of self-driving car development
  - Crash tests