CANvas: Fast and Inexpensive Automotive Network Mapping

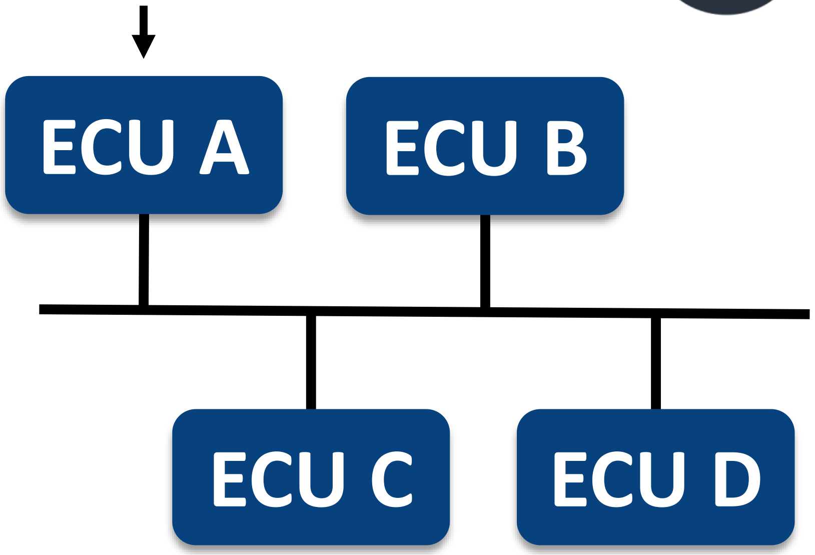
Sekar Kulandaivel, Tushar Goyal,

Arnav Kumar Agrawal, Vyas Sekar

Carnegie Mellon University

Do you know what’s going on in your car?

Electronic Control Unit (ECU)



ECU A

ECU B

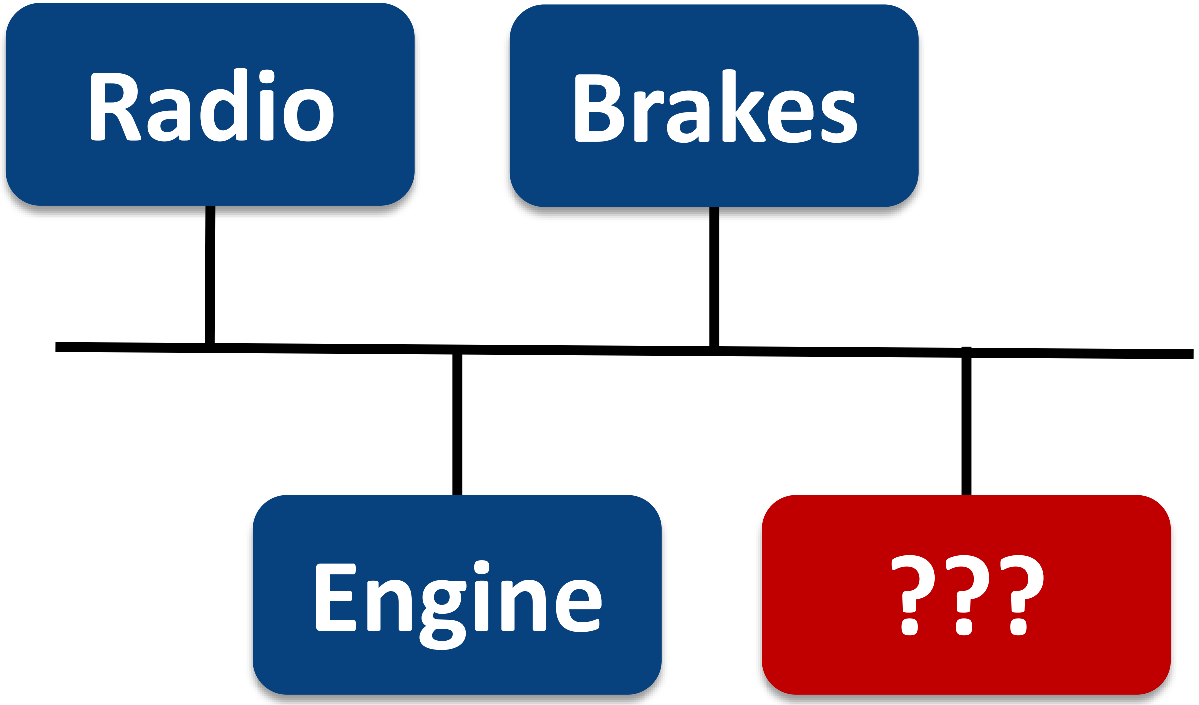
ECU C

ECU D

*Koscher et al. *IEEE S&P* ‘10
*Checkoway et al. *USENIX Security* ‘11

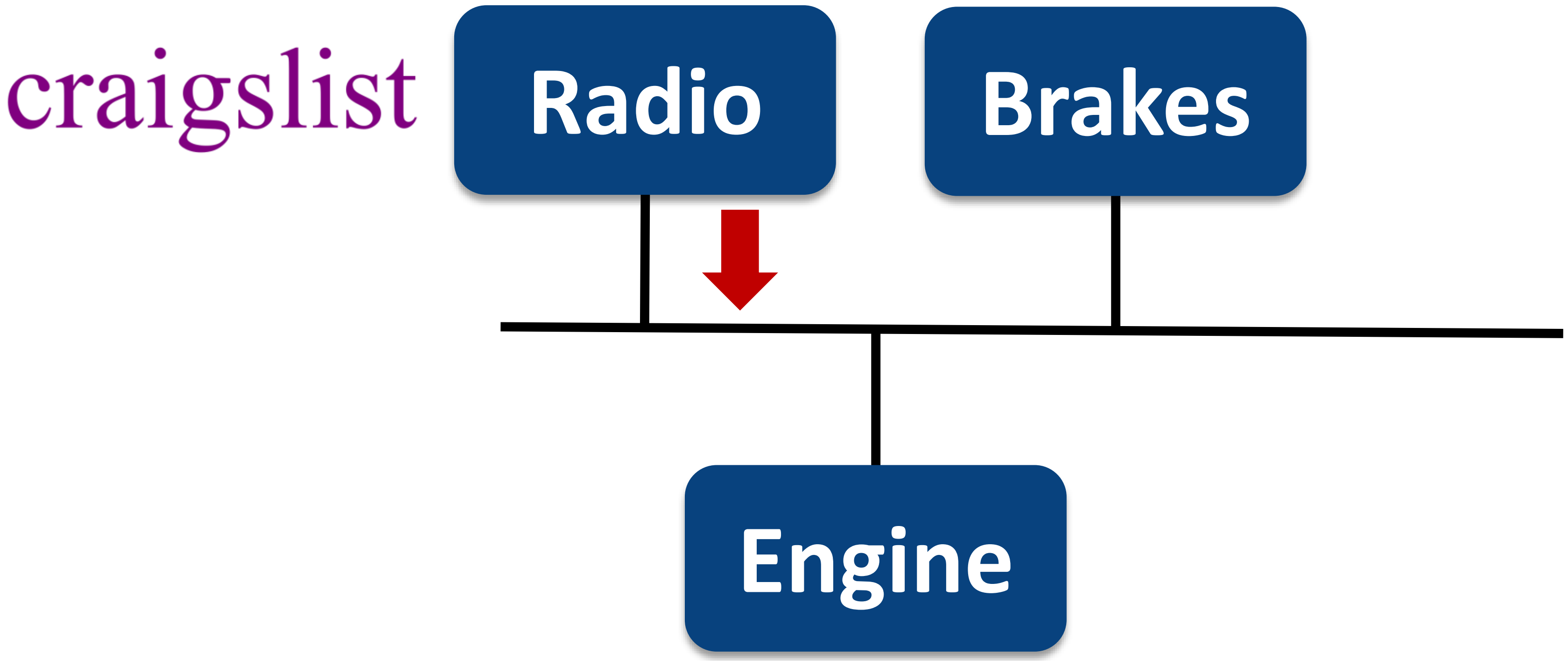
It’s important to know what’s going on inside your car

Scenario 1: the shady mechanic



Need to identify ECUs in the car

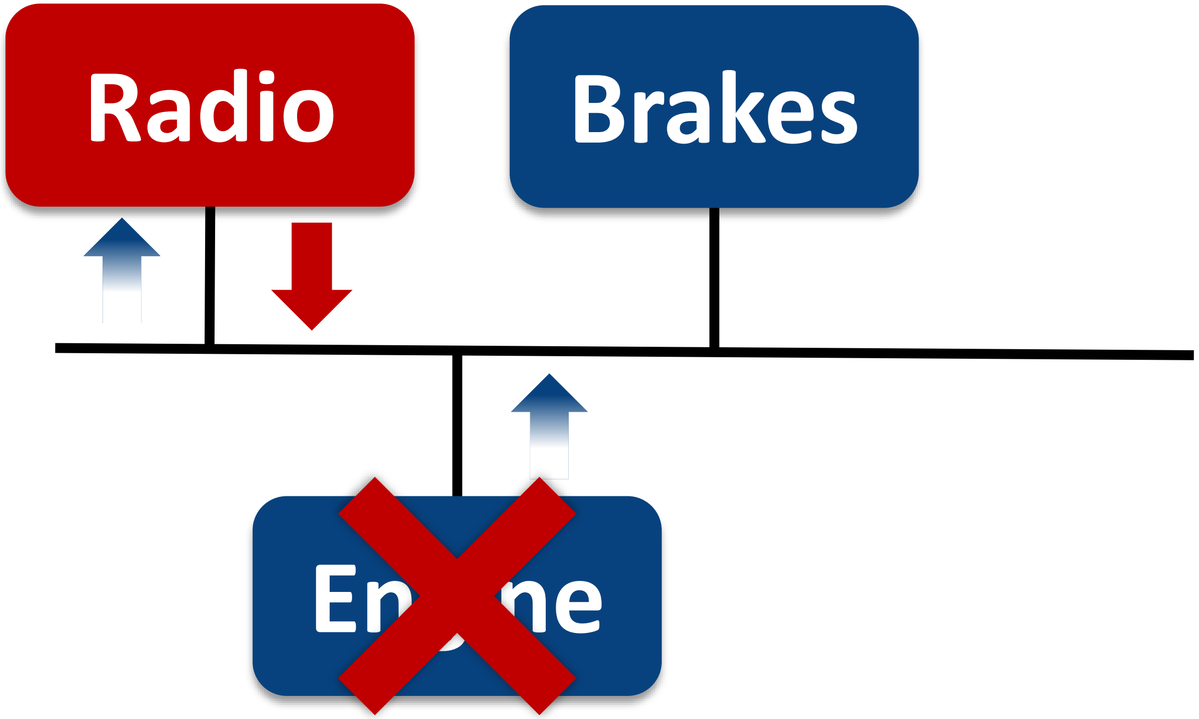
Scenario 2: the radio from Craigslist



Need to know who sends each message

Scenario 3: the shut-down attack

*Cho et al., *ACM CCS ‘16*



Radio

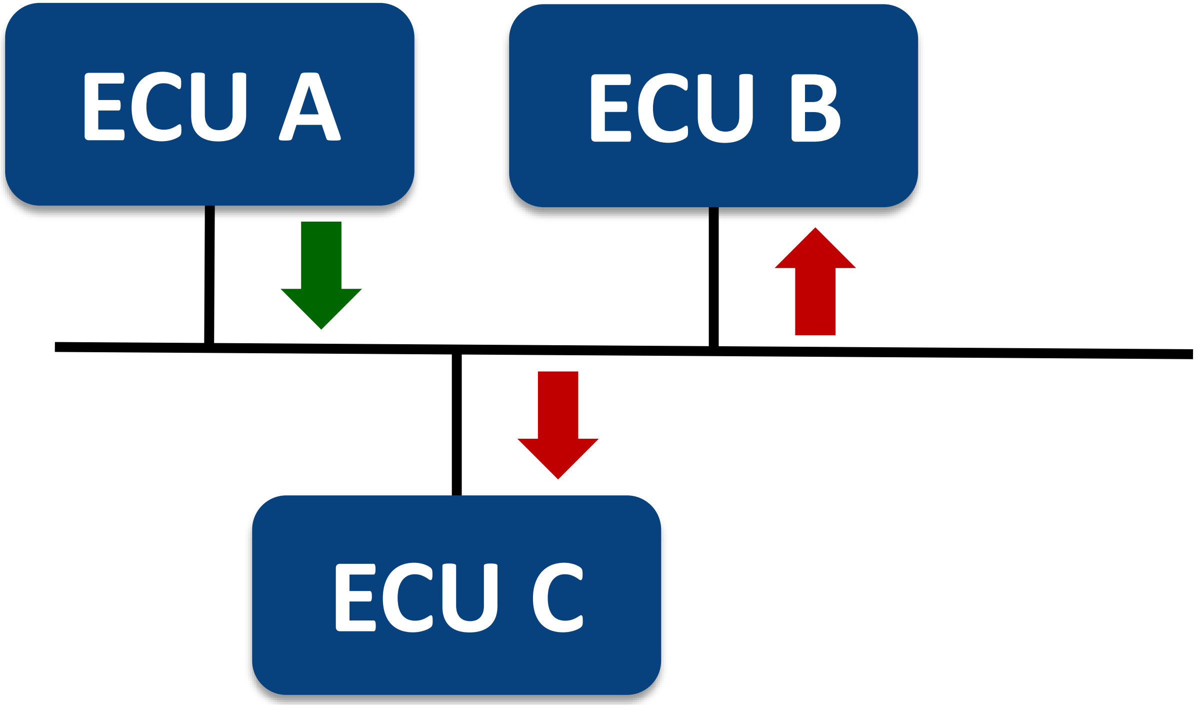
Brakes

Engine

Need to know who receives each message

We need an automotive network mapper

1. Identify ECUs
2. Identify message sender
3. Identify message receiver(s)



Requirements for a practical tool

Fast and inexpensive

Vehicle-agnostic

Minimally-intrusive

Non-destructive

Why not ask the automaker?

Confidential database file of messages

Online mechanic subscription

Network inside a car can change

CANvas in a nutshell

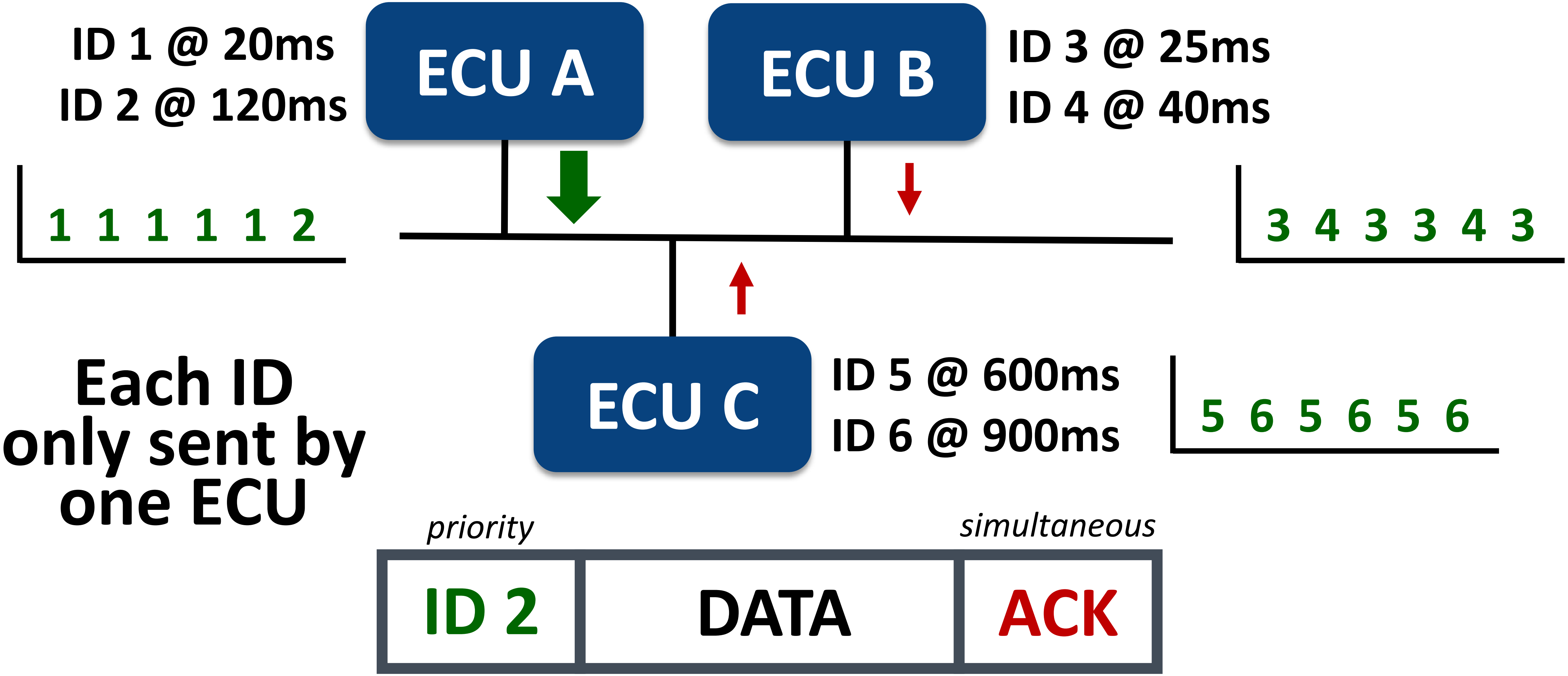
A network mapper for cars
that leverages message timing and
error-handling mechanism

Generates a network map in <30 minutes
with <$50 worth of hardware

Outline

- Motivating scenarios
- Background and mapping challenges
- System overview
- CANvas components
- Evaluation
- Conclusions

Controller Area Network (CAN) background



ID 1 @ 20ms
ID 2 @ 120ms

ECU A

1 1 1 1 1 2

ECU B

ID 3 @ 25ms
ID 4 @ 40ms

3 4 3 3 4 3

ECU C

ID 5 @ 600ms
ID 6 @ 900ms

5 6 5 6 5 6

Each ID only sent by one ECU

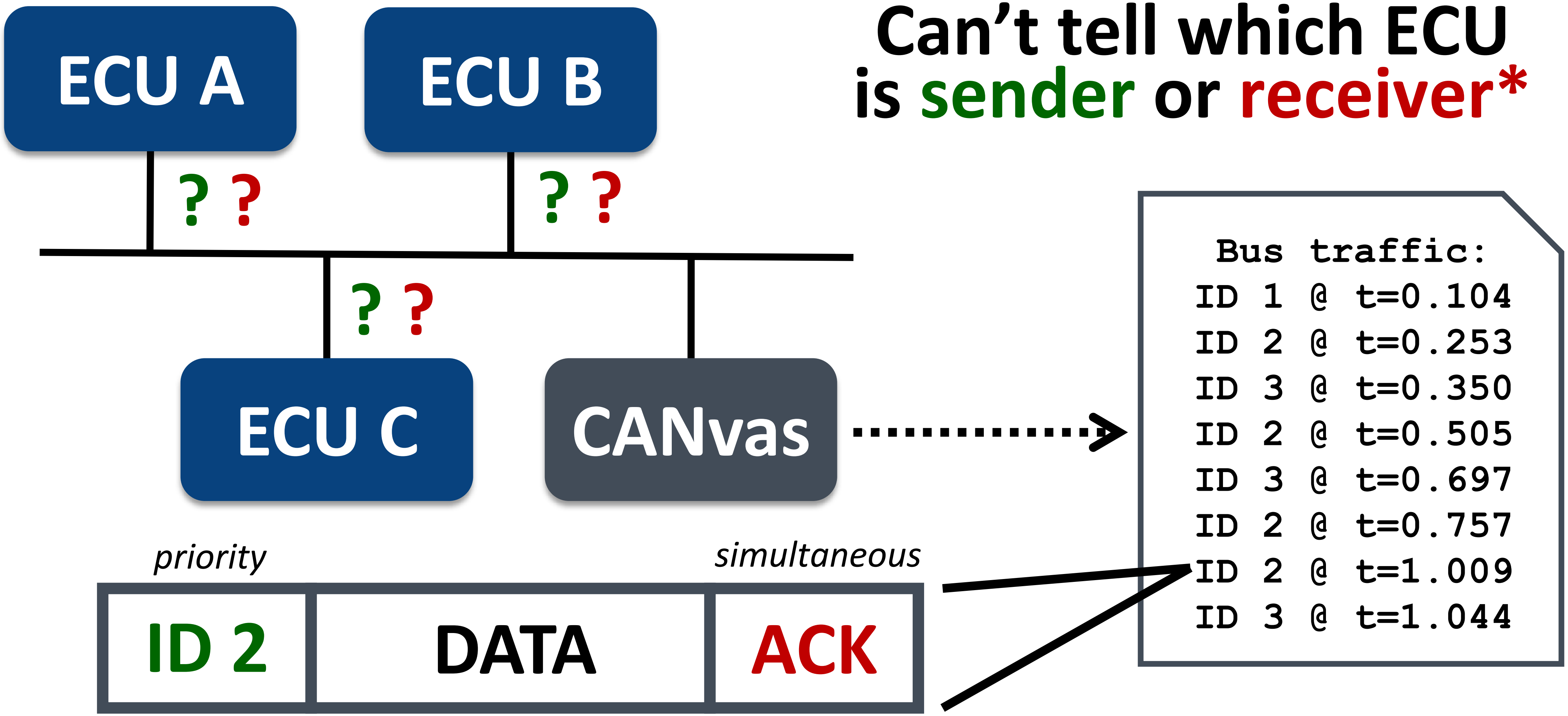
priority

simultaneous

ID 2	DATA	ACK

CAN makes mapping difficult

Can’t tell which ECU is sender or receiver*



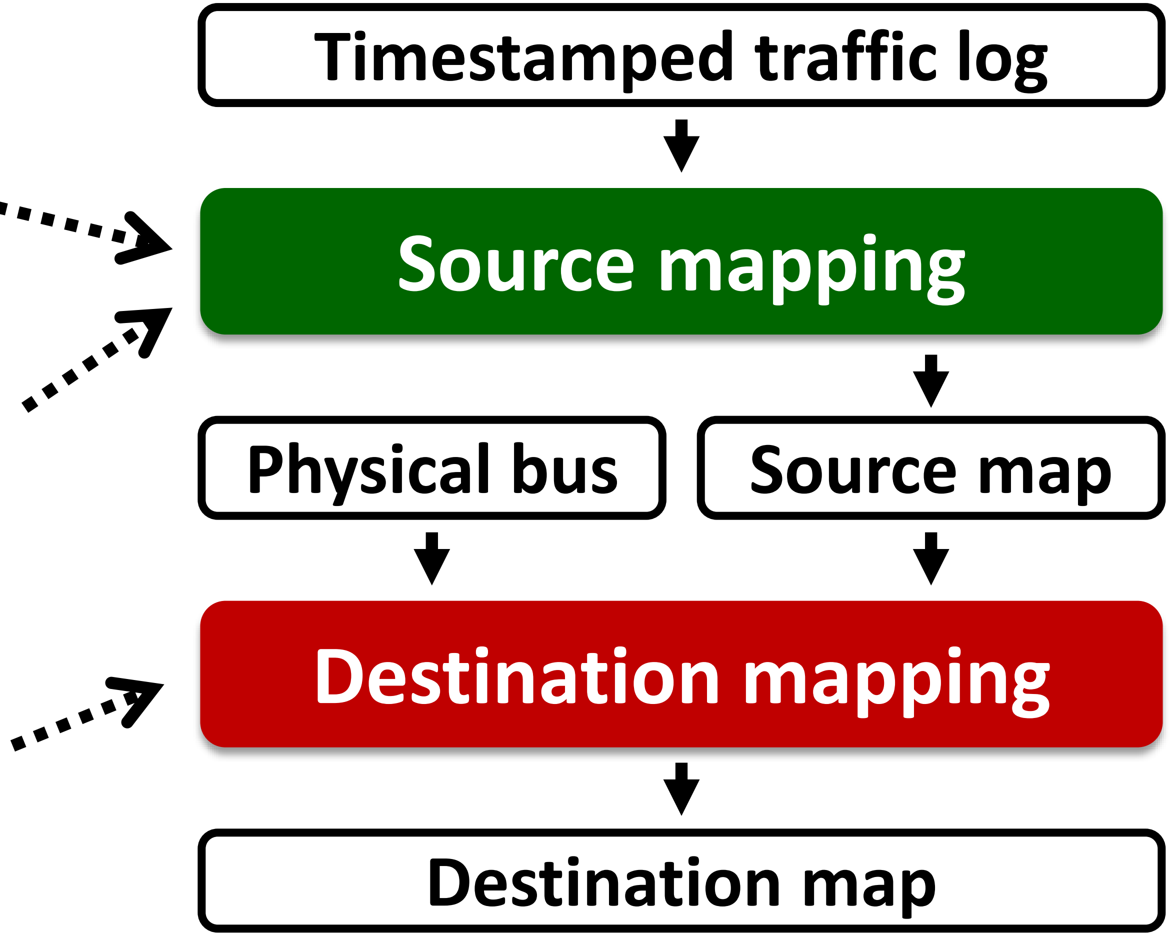
```
  Bus traffic:
ID 1 @ t=0.104
ID 2 @ t=0.253
ID 3 @ t=0.350
ID 2 @ t=0.505
ID 3 @ t=0.697
ID 2 @ t=0.757
ID 2 @ t=1.009
ID 3 @ t=1.044
```

Outline

- Motivating scenarios
- Background and mapping challenges
- System overview
- CANvas components
- Evaluation
- Conclusions

CANvas design overview

1. Identify ECUs
2. Identify message sender
3. Identify message receiver(s)



Timestamped traffic log

Source mapping

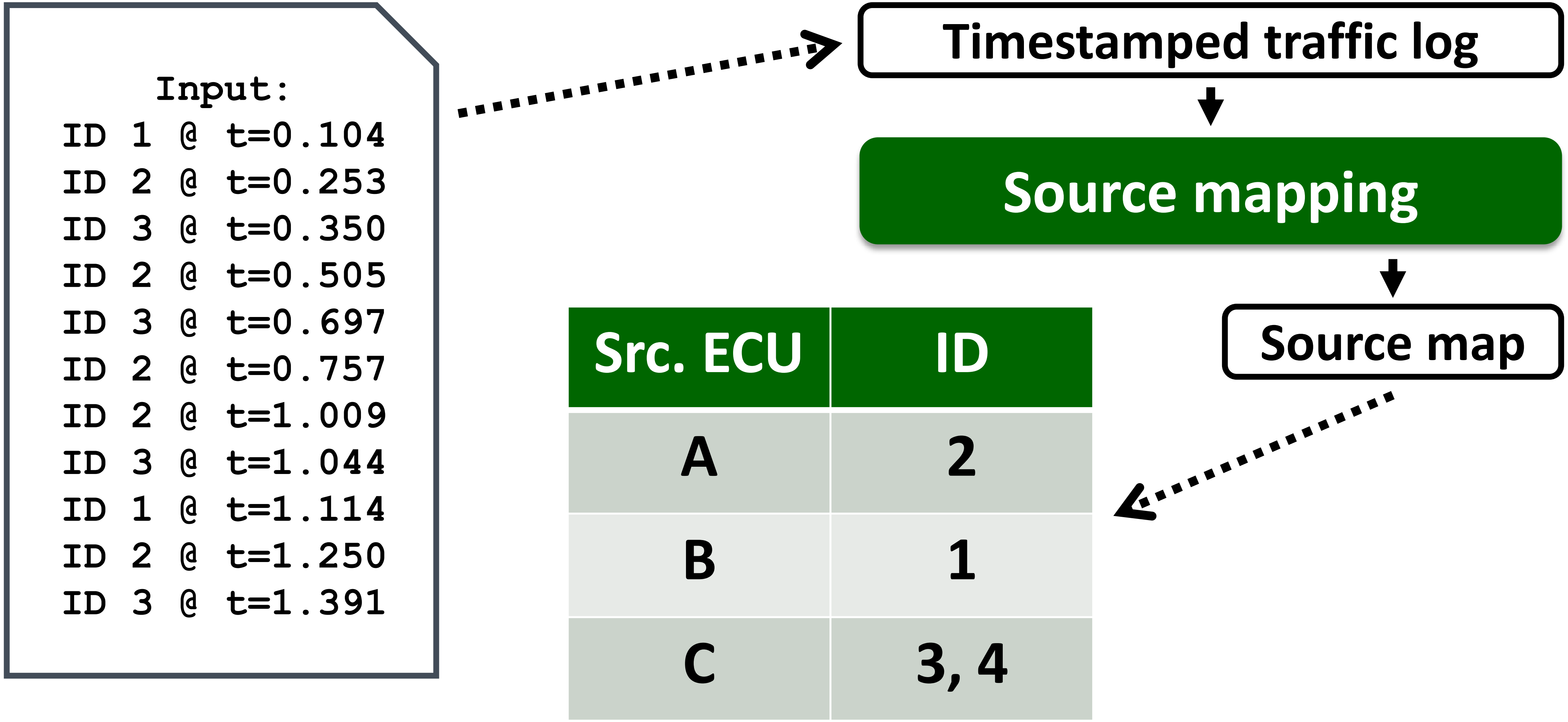
Physical bus

Source map

Destination mapping

Destination map

The source mapping problem



```
      Input:
ID 1 @ t=0.104
ID 2 @ t=0.253
ID 3 @ t=0.350
ID 2 @ t=0.505
ID 3 @ t=0.697
ID 2 @ t=0.757
ID 2 @ t=1.009
ID 3 @ t=1.044
ID 1 @ t=1.114
ID 2 @ t=1.250
ID 3 @ t=1.391
```

Src. ECU	ID
A	2
B	1
C	3, 4

Insight: clock offset as a unique identifier

*Cho et al., *USENIX Security '16*

Prior work for IDS

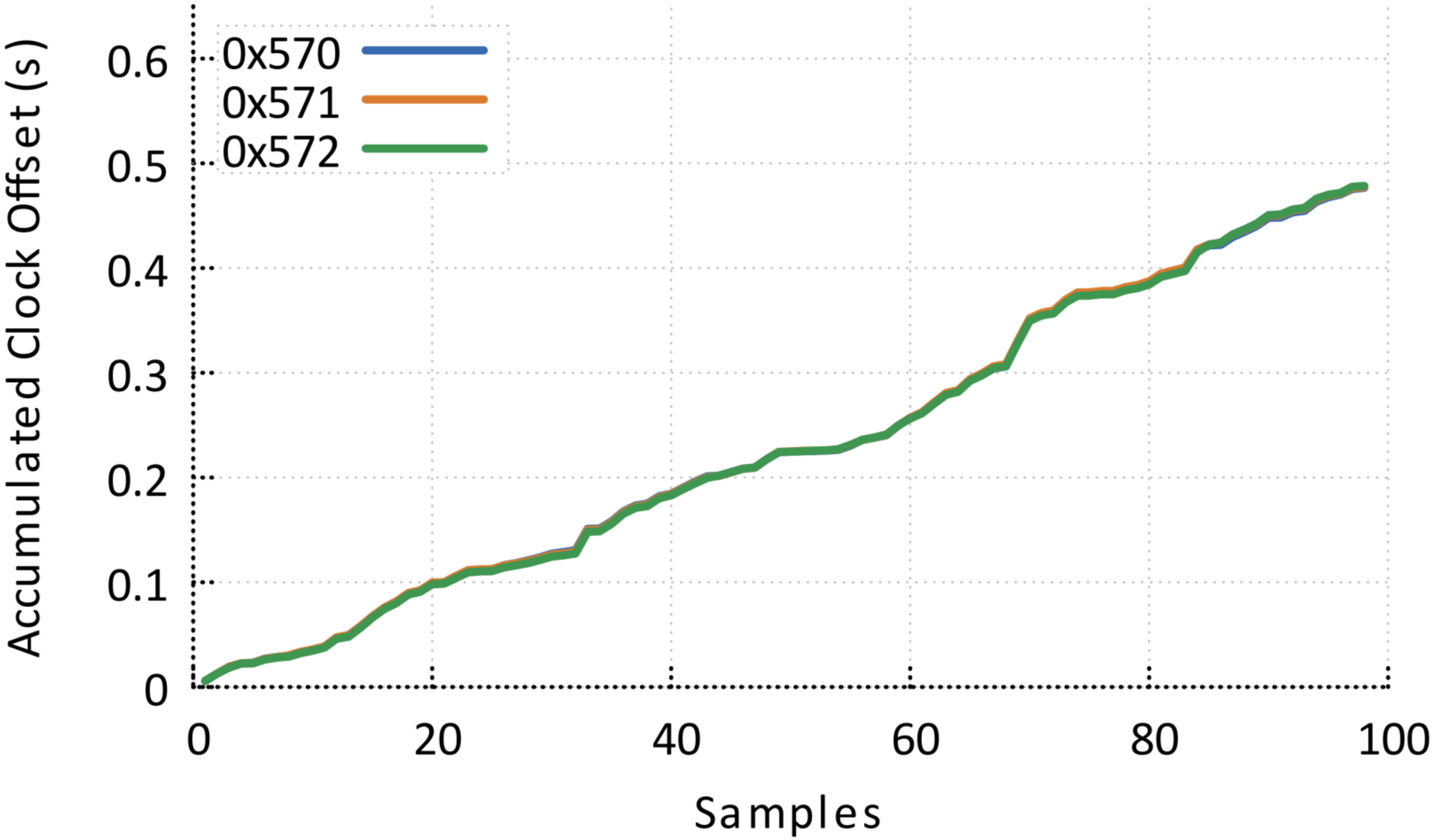
- Clock offset is unique
  - Track offset per ID

ECU X

ID 570 @ 1000ms

ID 571 @ 1000ms

ID 572 @ 1000ms



Limitations: prior work is not sufficient

Not robust to noise in the period

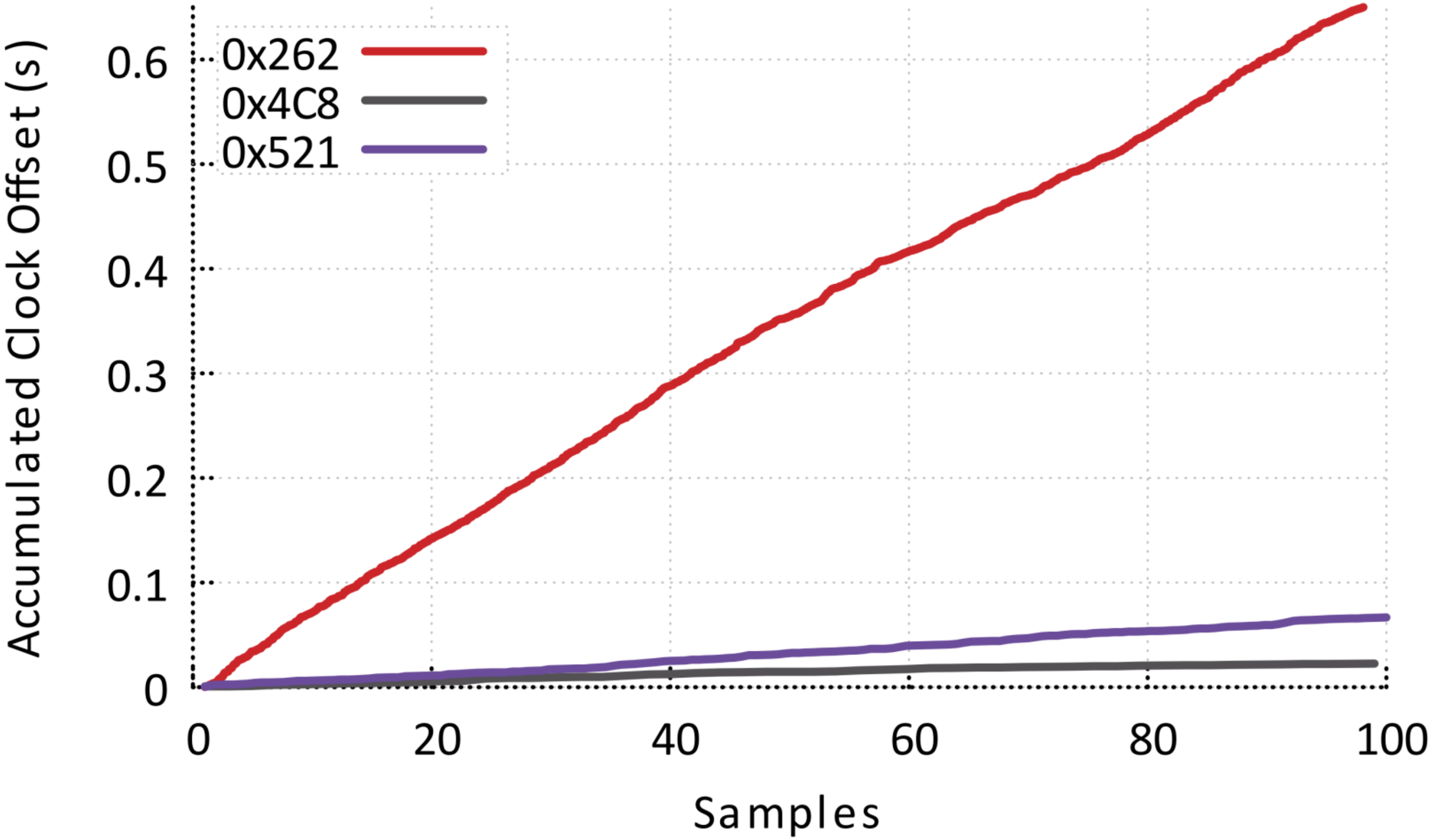
Period-dependent

ECU Y

ID 262 @ 20ms

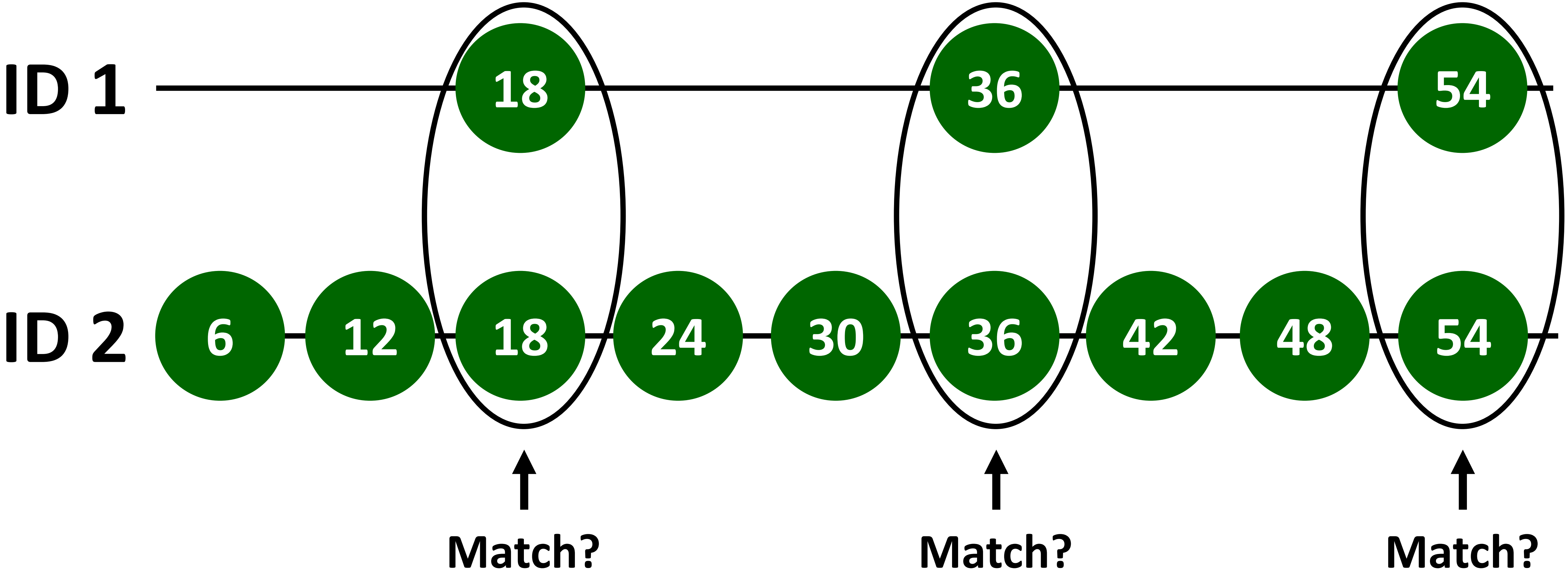
ID 4C8 @ 980ms

ID 521 @ 300ms



Idea: compare offset at hyper-period

Hyper-period removes period-dependence

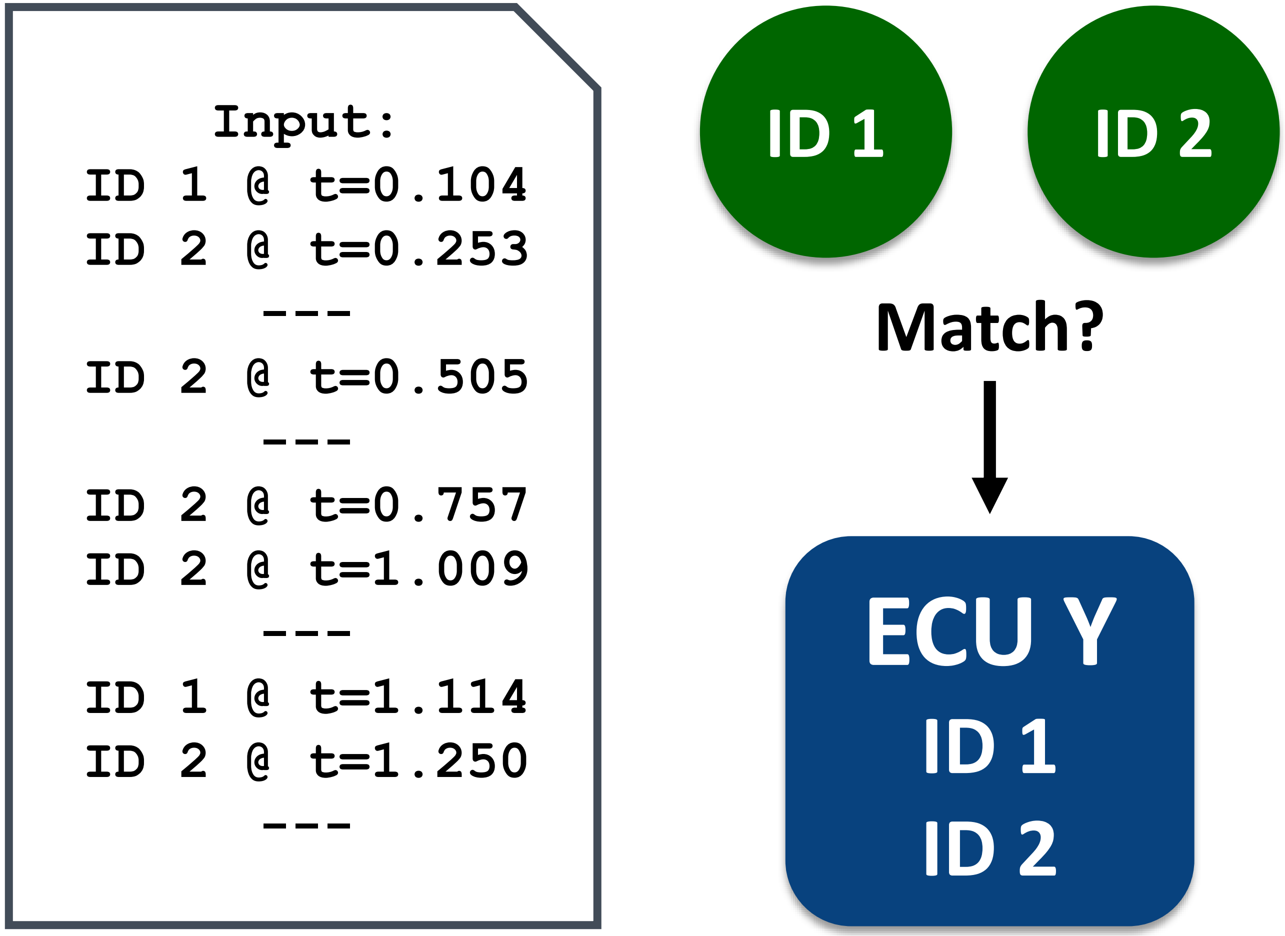


Approach: pairwise comparison over time

Hyper-period is period-independent

Measure over time to reduce effect of noise

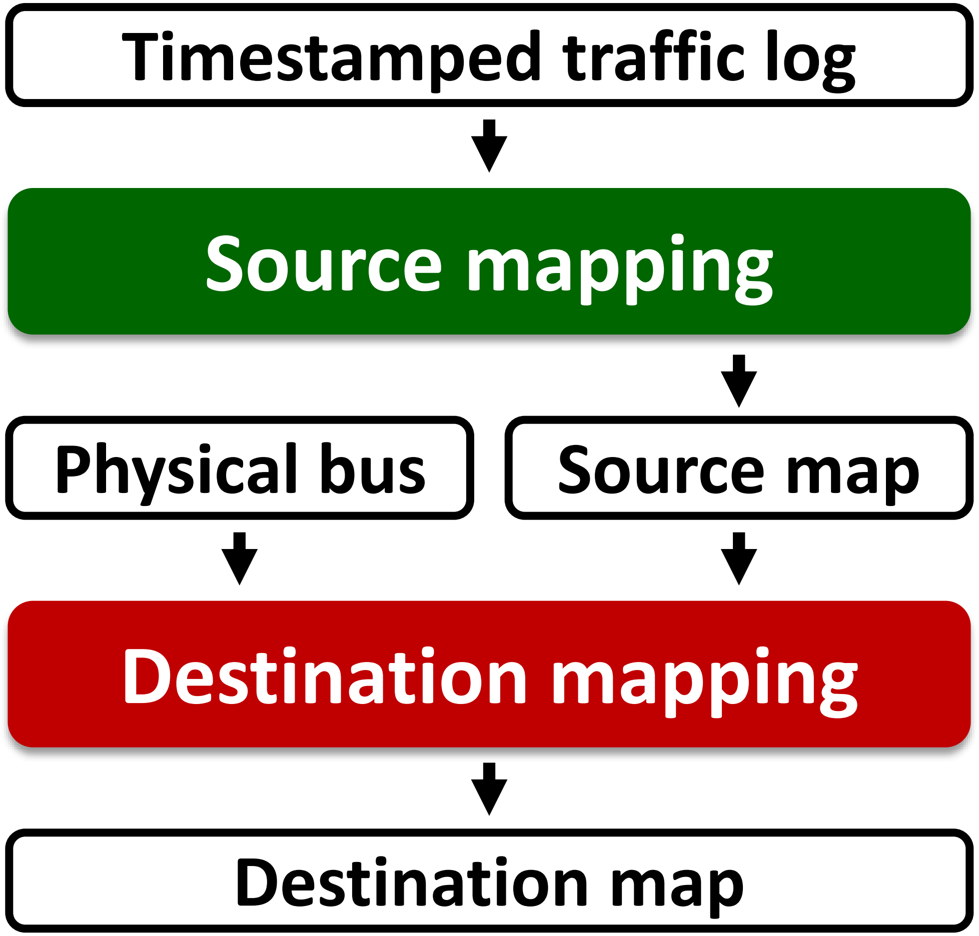
Practical challenges discussed in paper



```
       Input:
ID 1 @ t=0.104
ID 2 @ t=0.253
        ---
ID 2 @ t=0.505
        ---
ID 2 @ t=0.757
ID 2 @ t=1.009
        ---
ID 1 @ t=1.114
ID 2 @ t=1.250
        ---
```

CANvas design overview

1. Identify ECUs
2. Identify message sender
3. Identify message receiver(s)



Timestamped traffic log

Source mapping

Physical bus

Source map

Destination mapping

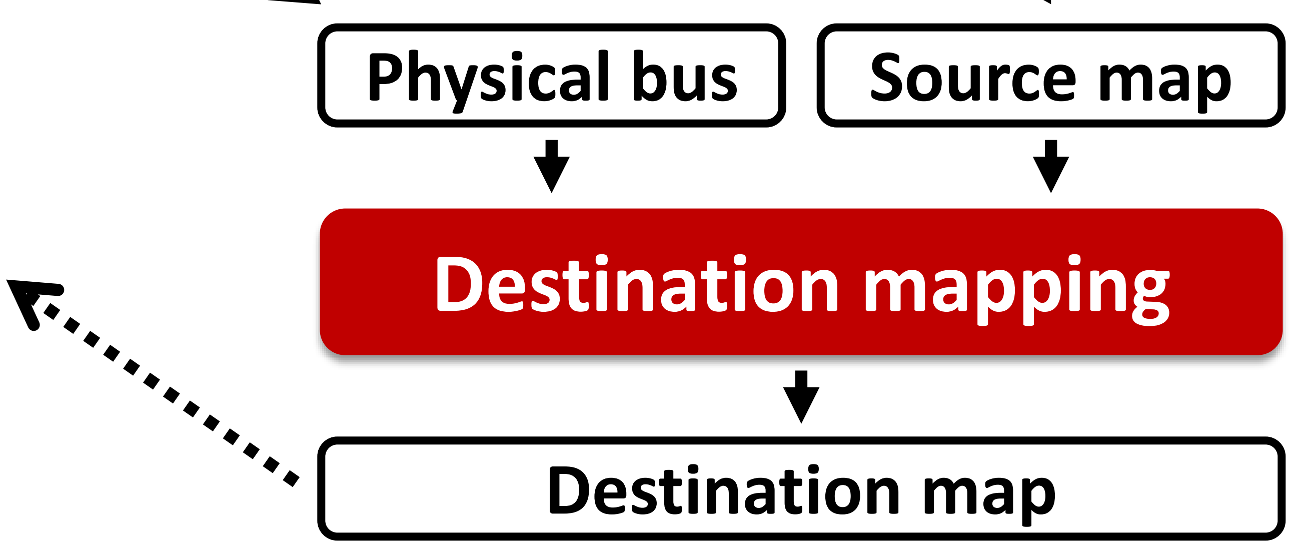
Destination map

The destination mapping problem

1
8
Diagnostics port
9
16

ID	Dst. ECUs
1	A, C
2	B, C
3	B

Src. ECU	ID
A	2
B	1
C	3

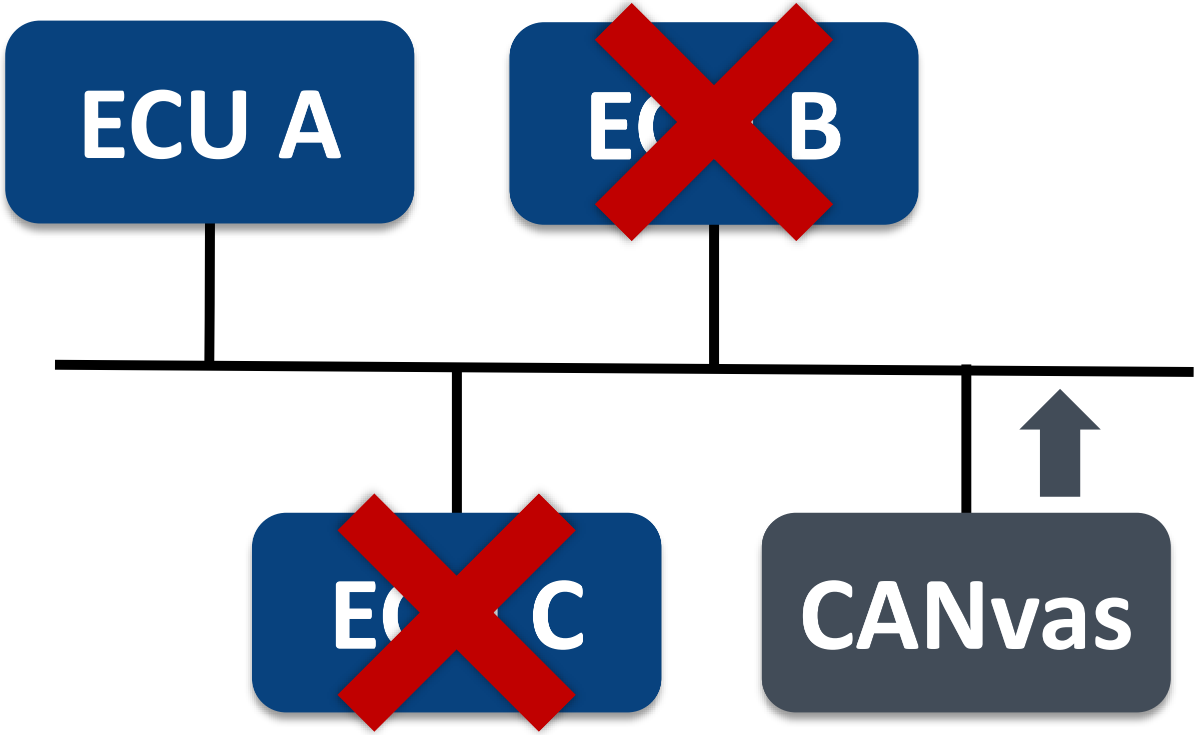

Physical bus

Source map

Destination mapping

Destination map

Approach: isolate each ECU



Isolate an ECU to guarantee who sent ACK

ACK indicates some ECU received

ID	Dst. ECUs
1	???

ID	Dst. ECUs
1	A

Insight: shut-down via error-handling exploit

*Cho et al., *ACM CCS ‘16*

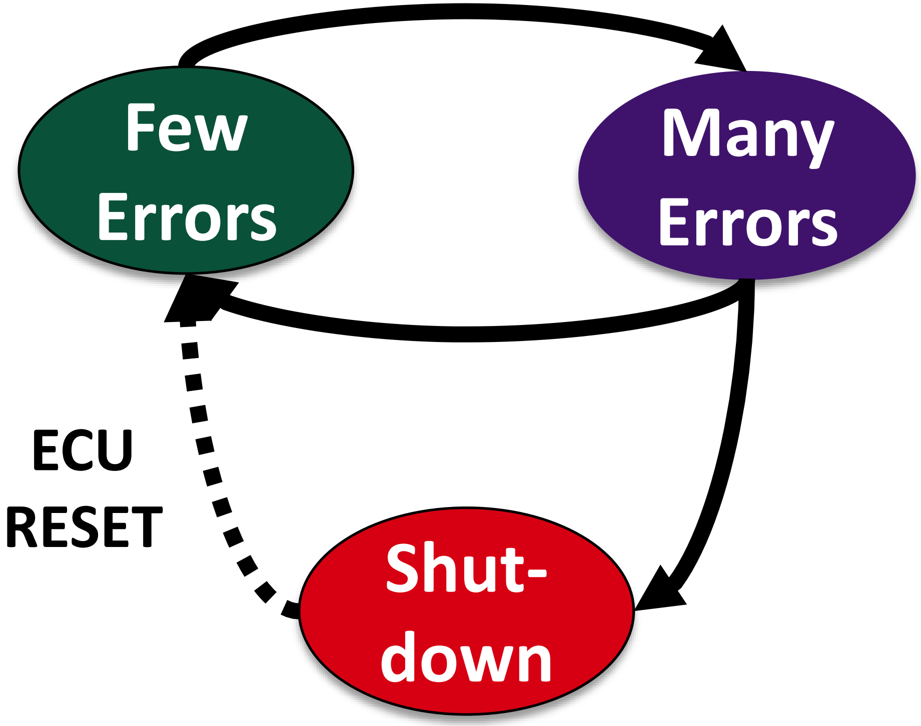
Prior work for a DoS attack

- Exploit error-handling by causing errors

Not intended to be robust – attack needs just one success

Refer to paper for limitations and our idea for isolation

After too many errors, an ECU will shut-down!



Outline

- Motivating scenarios
- Background and mapping challenges
- System overview
- CANvas components
- Evaluation
- Conclusions

Evaluation setup

2009 Toyota Prius

2017 Ford Focus

Junkyard ECUs

Ford Engine ECUs

Synthetic topologies

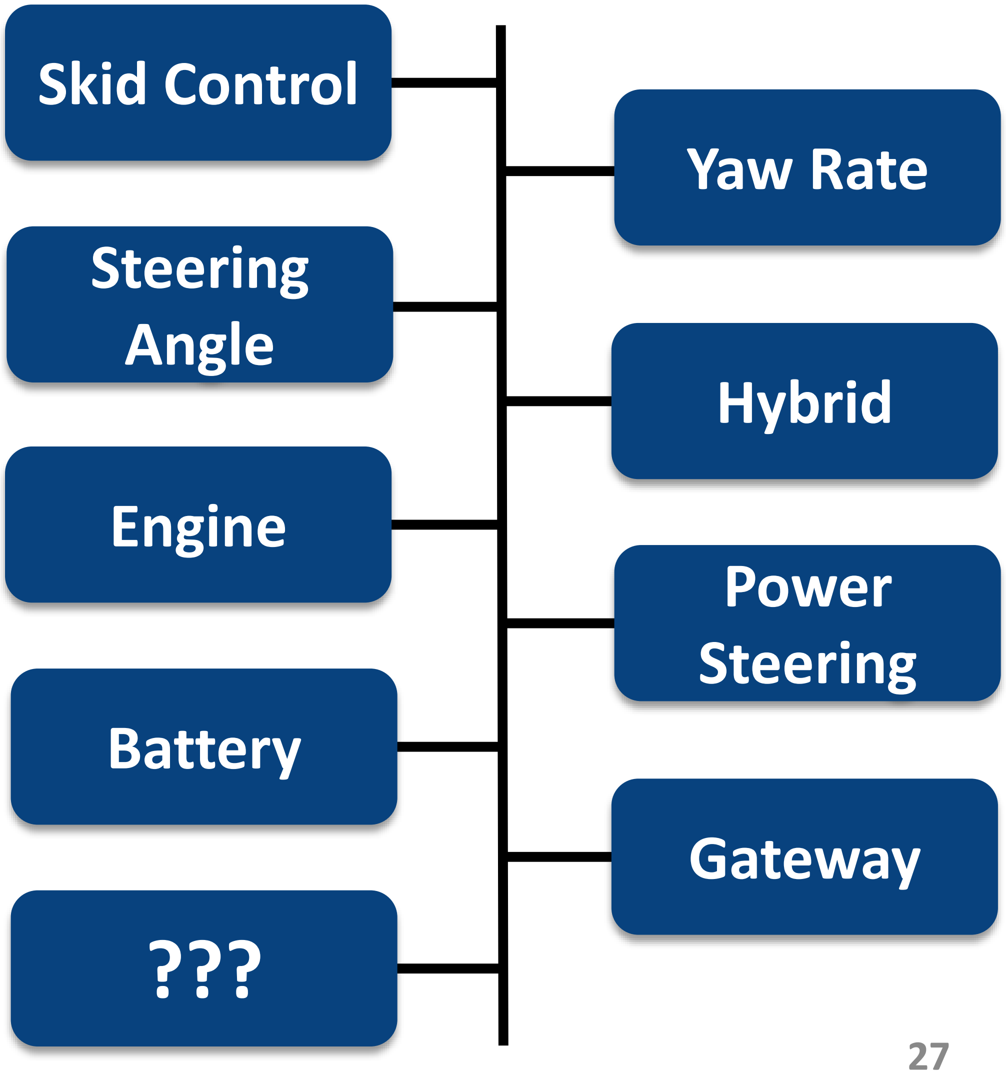
Arduino Due

Key takeaways

Fast	✓	<30 minutes
Inexpensive	✓	<$50
Vehicle-agnostic	✓	Standard CAN
Minimally-intrusive	✓	OBD-II port
Non-destructive	✓	No dash lights

Finding an unexpected ECU in a ‘09 Prius

ECU installed during a past vehicle modification



Skid Control

Steering Angle

Engine

Battery

???

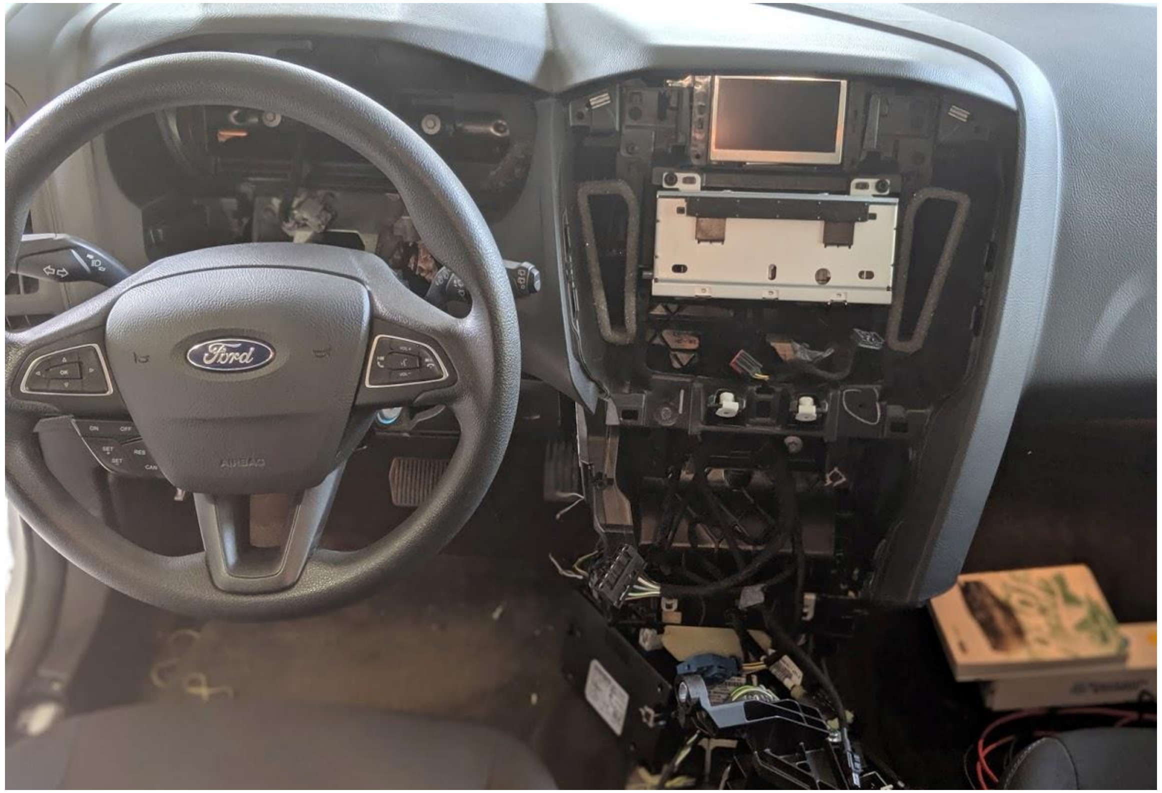
Yaw Rate

Hybrid

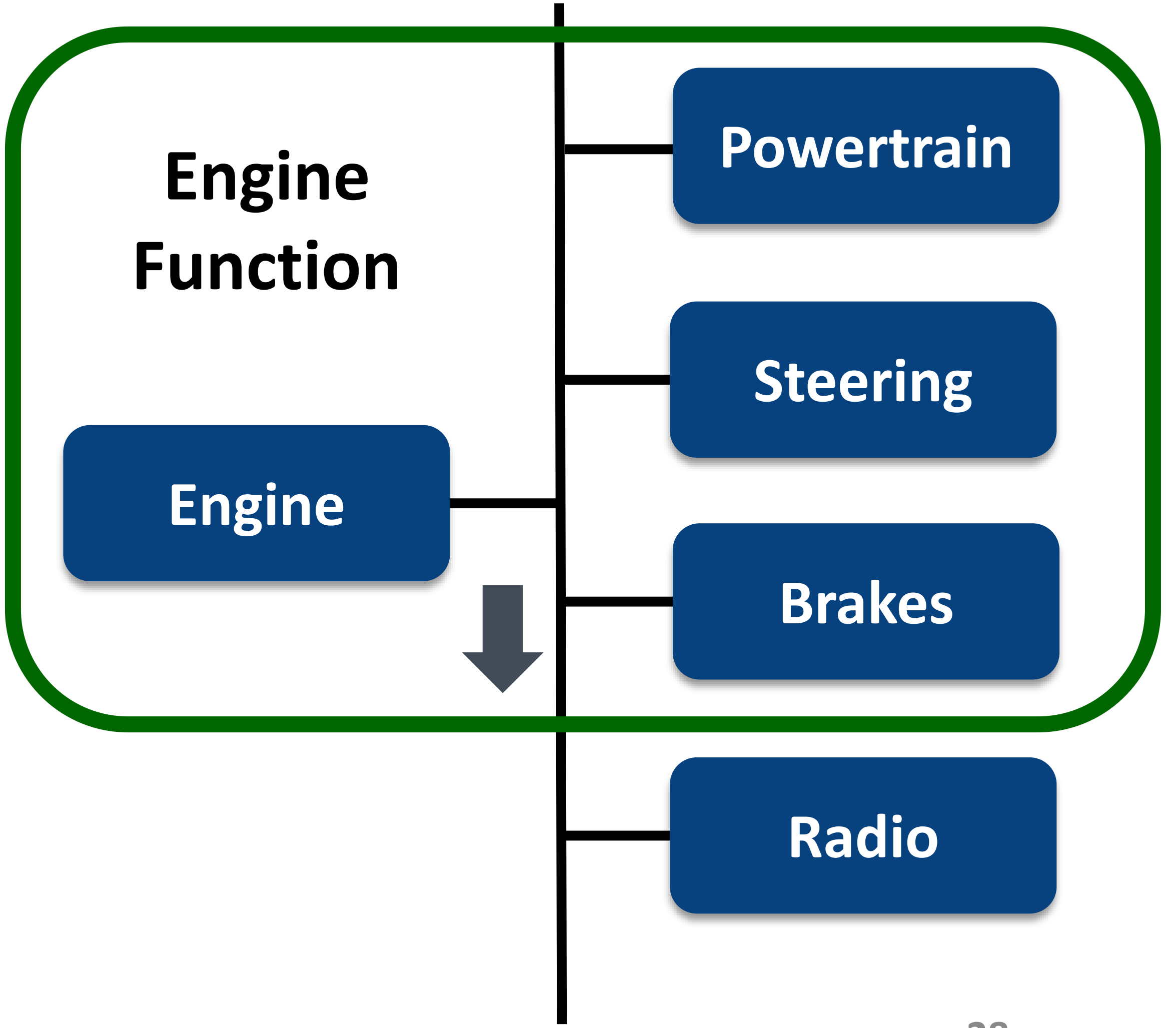
Power Steering

Gateway

‘17 Focus ECUs enable the shut-down attack



Both Prius and Focus had no filter on what messages an ECU could receive



CANvas limitations

Adversarial evasion

Timing-aware attacker

Intentional timing alteration

Avoiding permanent damage

Resetting dash lights

Limp-home mode

Multiple CAN buses

Accessing unexposed buses

Message acceptance filter

Vendor-specific approaches

Non-transmitting ECUs

Conclusions

- Network inside cars can change
- **CANvas**: a network mapper that tells us what’s going on in a car
- Mapping CAN is non-trivial → lack of source or destination info
- Prior work did not solve mapping goals
- A fast and inexpensive design focused on practicality
- Real-world demonstration on two vehicles
- Serves as a basis for many other security applications

https://github.com/sekarkulandaivel/canvas