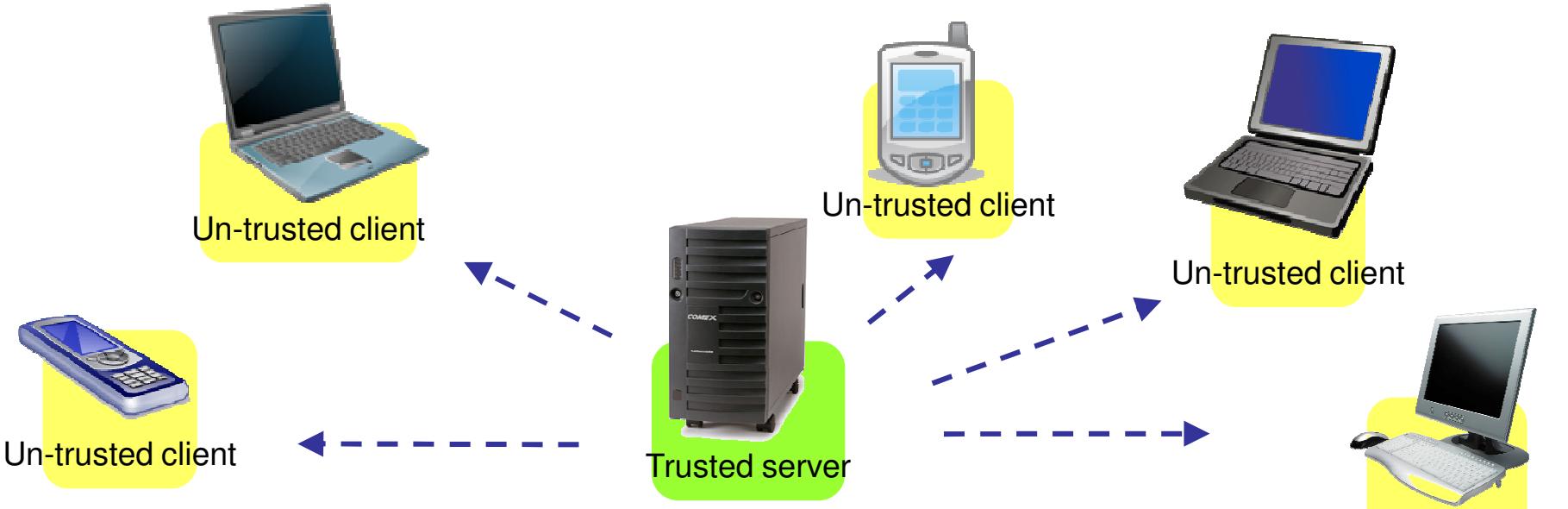
Barrier Slicing for Remote Software Trusting

***Ceccato Mariano*[1], *Mila Dalla Preda*[2],**
***Jasvir Nagra*[2],**
***Christian Collberg*[3], *Paolo Tonella*[1]**

[1]*Fondazione Bruno Kessler-IRST, Trento, Italy*
[2]*University of Trento, Italy*
[3]*University of Arizona, USA*

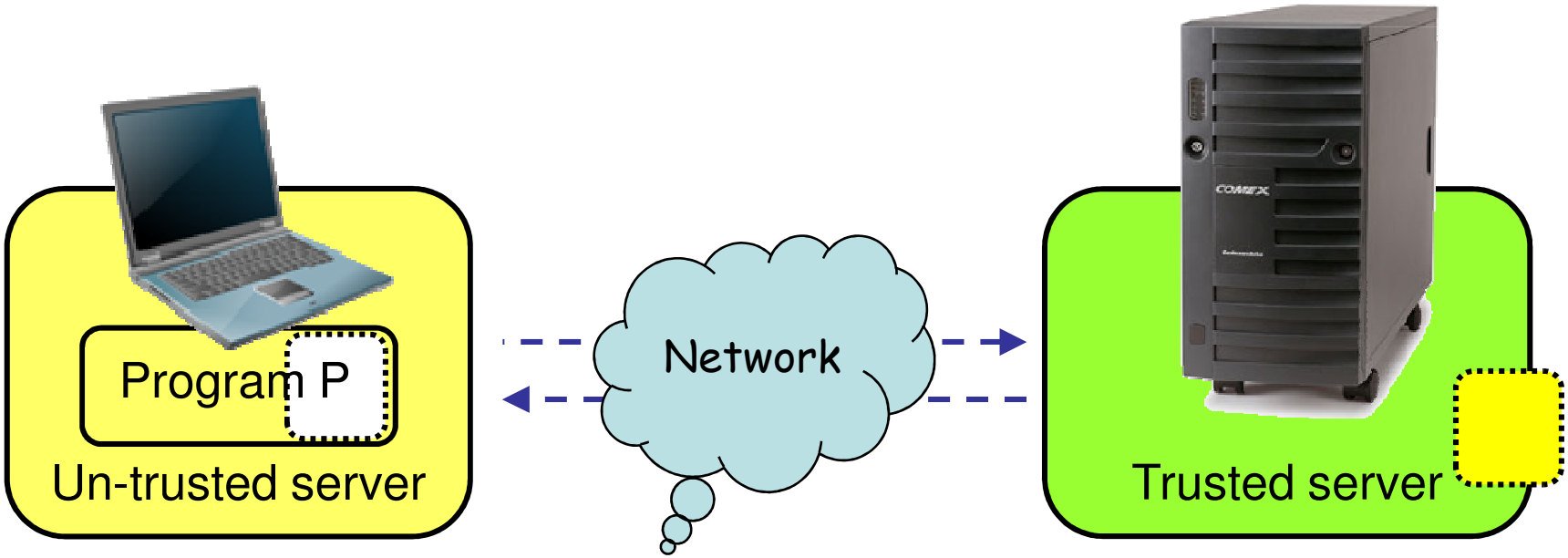
Remote software trusting

- *Remote software authentication*: ensuring a trusted machine (server) that an un-trusted host (client) is running a “healthy” version of a program;
- The server is willing to deliver a given services only to clients that prove to be “healthy”;
  - The program is unadulterated.
  - It is executed on top of unadulterated HW/SW.
  - The execution process is not manipulated externally.



Approach

- Remove a portion of the program to protect and run it on the server.
  - Trade off between security and performances



Program slice

- Set of variables that we are interested in protecting.
- We remove those variable from the client.
- The (executable) slice is replicated into the server where it can be executed safely.

```
1   time2 = System.currentTimeMillis();
2   double delta = speed * (time2 - time);
3   x = x + delta * cos(direction);
4   y = y + delta * sin(direction);
5   Server.sendPosition(x,y);
6   if (track.isInBox(x, y)){
7           gas = maxGas;
8           lastFuel = time2;
9   }
10  else {
11          gas = maxGas - (int) (time2-lastFuel);
12          if (gas < 0) {
13                  gas = 0;
14                  if (speed > maxSpeed /10)
15                          speed = maxSpeed /10;
16                  else if (speed < minSpeed/10)
17                          speed = minSpeed/10;
            }
    }
18  time = time2;
```

Safe variables: barrier slice

- Subset of variables that can not modified by the user, otherwise either:
  - the client would receive a not-usable service, or
  - the server would notice it (using assertions)
- They can be used as **barriers** and block the dependency propagation when slicing (Krinke, scam 2003)

```
1   time2 = System.currentTimeMillis();
2   double delta = speed * (time2 - time);
3   x = x + delta * cos(direction);
4   y = y + delta * sin(direction);
5   Server.sendPosition(x,y);
6   if (track.isInBox(x, y)){
7           gas = maxGas;
8           lastFuel = time2;
9   }
10  else {
11          gas = maxGas - (int) (time2-lastFuel);
12          if (gas < 0){
13                  gas = 0;
14                  if (speed > maxSpeed /10)
15                          speed = maxSpeed /10;
16                  else if (speed < minSpeed/10)
17                          speed = minSpeed/10;
            }
    }
18  time = time2;
```

safe

un-safe

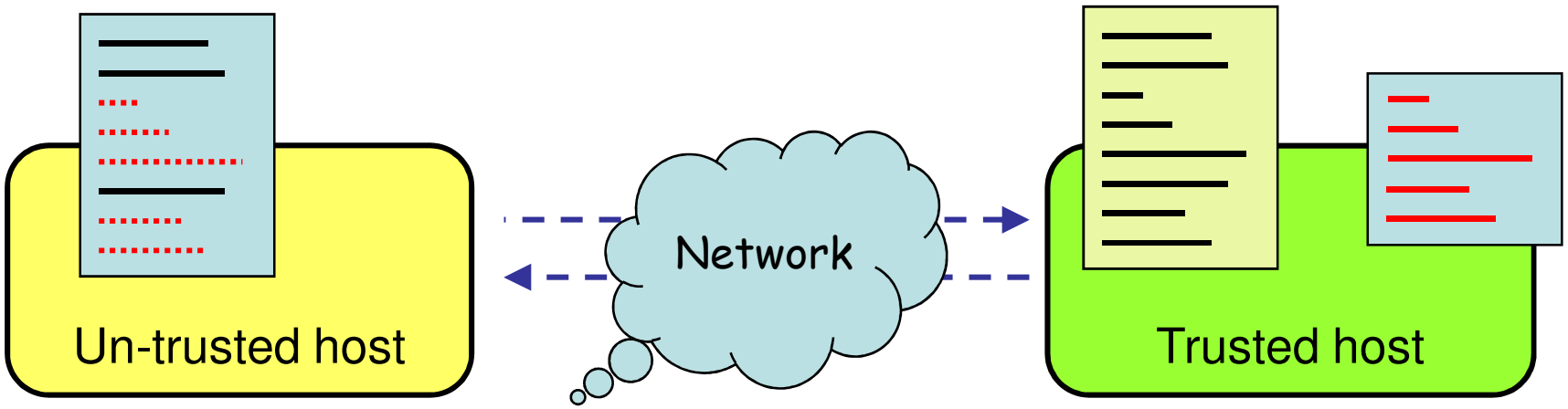
Program transformation

Un-trusted host:

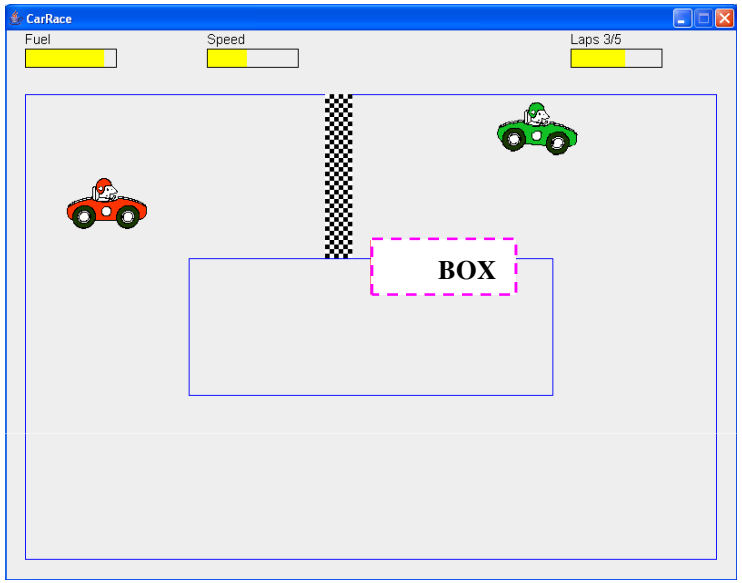
- X ∈ un-safe
- X **uses** are removed from the program;
- They are replaced by a query to get the actual value over the network;
- X **defs** are replaced by synchronization statements.
- Some optimizations…

Trusted host:

- A barrier-slice is run for each served host;
- Client validity is continuously verified (assertions);
- X values are provided as required;
- Synchronization with the un-trusted hosts.



Example: CarRace



- Position
- Number of Laps
- Fuel
- Speed

Original client	Slice	Barrier slice
858	185	120 (-65)
	22%	14% (-35%)

	Regular messages	Trust messaged	Increase
Sent	1174	5910	5.03
Received	1172	5910	5.04

Ongoing works

- Integrating the monitor with the slice approach to improve performances;
- Apply the barrier slicing to bigger test cases to perform overhead measurements;
- Integrate the approach with secure hardware;
- Automatic identification of the secure and un-secure variables.