TickTalk: Timing API for Federated Cyber-physical Systems

- Bob Iannucci (CMU), Carlee Joe-Wong (CMU), **Aviral Shrivastava** (ASU), Jonathan Aldrich (CMU)
- Carnegie Mellon University and Arizona State University
- http://ccsg.ece.cmu.edu/wp/index.php/home/ticktalk/
- bob@sv.cmu.edu

- CNS-1646235 (CMU), CNS-1645578 (ASU)

- June 2 (Session 2, Presentation 6)

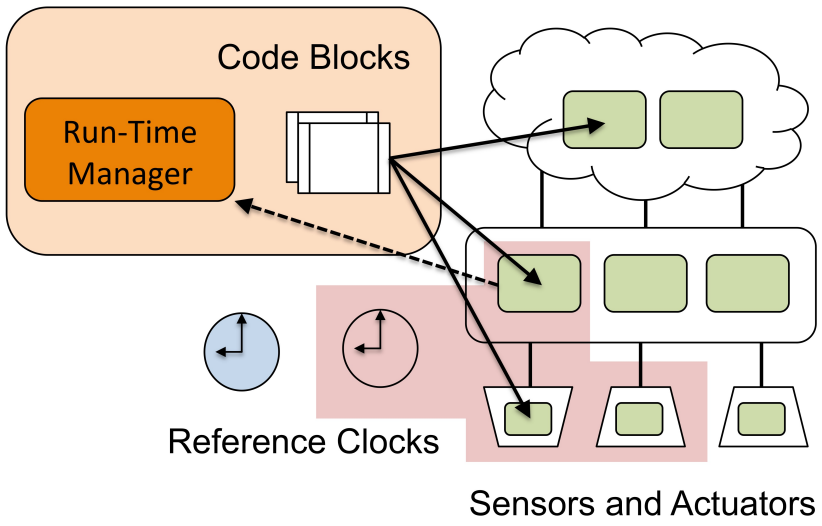
TickTalk: Timing API for Federated Cyber-physical Systems

The economic **potential** of large-scale CPS (*e.g.,* smart cities and environments) will be enabled in part through simplification of programming (like app development by non-specialists).

The need to meet the timing specifications makes programming large-scale distributed CPS **difficult**.

Proposal:

- Create a **programming language** that abstracts timing, timing-fault handling and related power management issues
- Develop **hardware extensions** that support low-power sync, timing-related power reporting, and multi-tenancy
- Create an **end-to-end demonstration** including a compiler and runtime; deploy in a real-world testbed



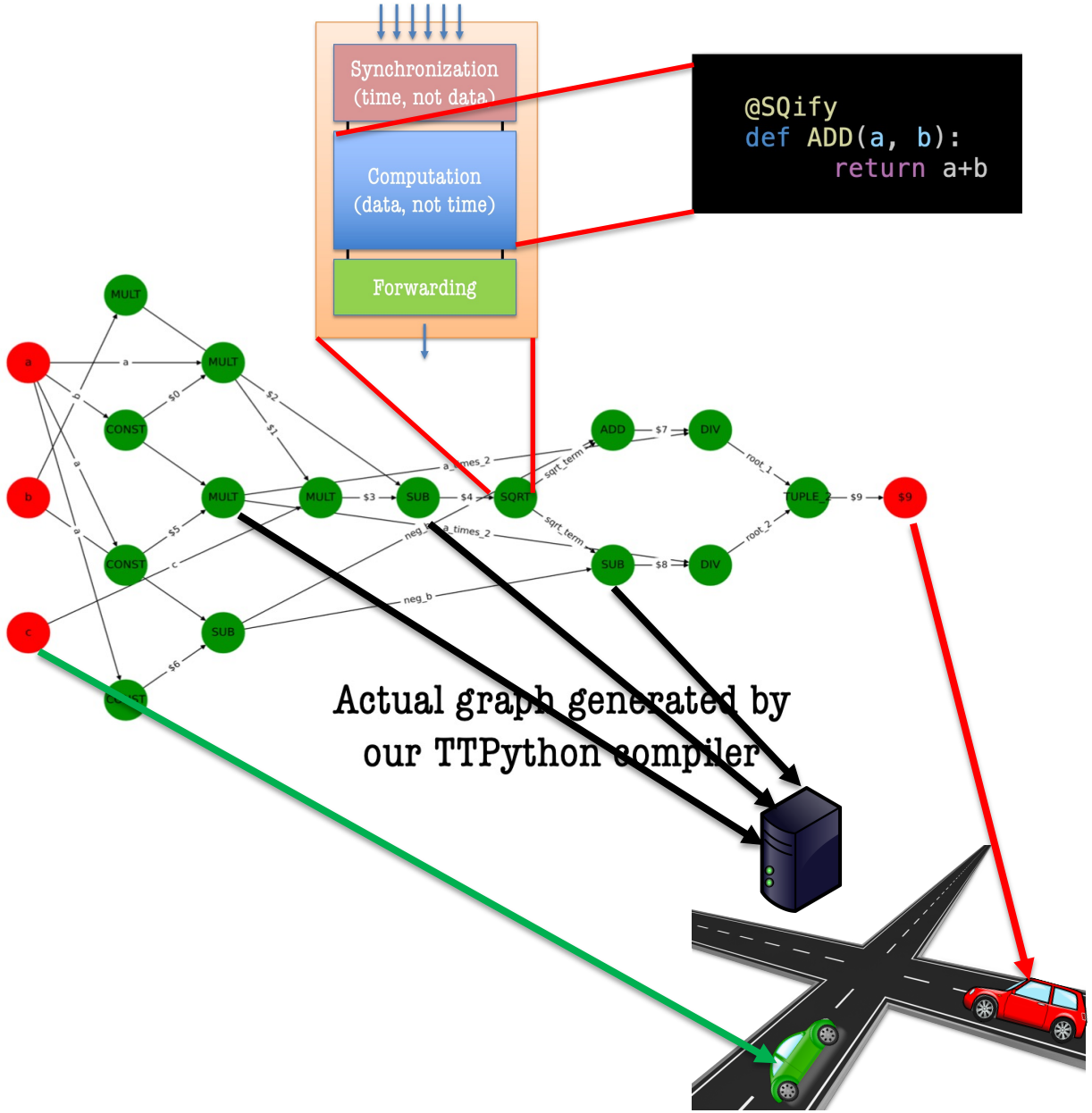
TTPython – Time-sensitive Macro-programming Language

```python
@GRAPHify
def main(a, b, c):
    with TTClock("local_root") as CLOCK:
        with TTPlanB(planB_handler):
            with TTDeadline(CLOCK, 500):
                sqrt_term = SQRT((b * b) - CONST(a, const=4) * a * c)
                a_times_2 = CONST(a, const=2) * a
                neg_b = CONST(a, const=0) - b
                root_1 = (neg_b + sqrt_term) / a_times_2
                root_2 = (neg_b - sqrt_term) / a_times_2
                return TUPLE_2(root_1, root_2)
```

STEP 1: The programmer adds the @SQify annotation to function definitions. This says "expect the inputs to arrive as time-tagged values, but please compute the function body without my having to think about that."

STEP 2: The programmer writes a Python-like program that is made up of calls to these SQified functions. Normal infix operations can be used for basic arithmetic operations. Importantly, time annotations like deadlines can be added.

Under the covers: the TTCompiler translates this program into a dataflow graph made up of instances of the SQified functions hooked together with arcs and triggered by tokens.



Demo: Traffic Intersection for Autonomous Cars

- Crossroads Algorithm for Managing Traffic Intersection for Autonomous Vehicles.
- When vehicles approach the intersection, they send their position and velocity to the intersection manager.
- Intersection manager knows the schedule of all the vehicles crossing the intersection, and assigns a speed to the incoming vehicle for safe and efficient crossing.
- Crossroads algorithm considers the computation and communication delays as safety buffer around the vehicle.

Advantages of our approach

- Macro-programming – Program the whole system as one application
- Explicitly the specify timing constraints – using the “with” decorator
- Timing constraints can be distributed
- Automatic clock synchronization, timestamp translation
- Dataflow execution and token passing under the hood
- Portable timing and code