Using Integer Clocks to Verify Timing-Sync Sensor Network Protocol

Xiaowan Huang, Anu Singh, Scott Smolka
Stony Brook University

Introduction

- Goal
  - Model check the TPSN protocol.
- Tool
  - The Uppaal model checker for real-time systems modeled as networks of timed automata.
- Problem
  - Constraints of clocks in timed automata: a clock can only be assigned a constant value.
  - Requirement of arithmetic operations of clock values in the TPSN protocol.

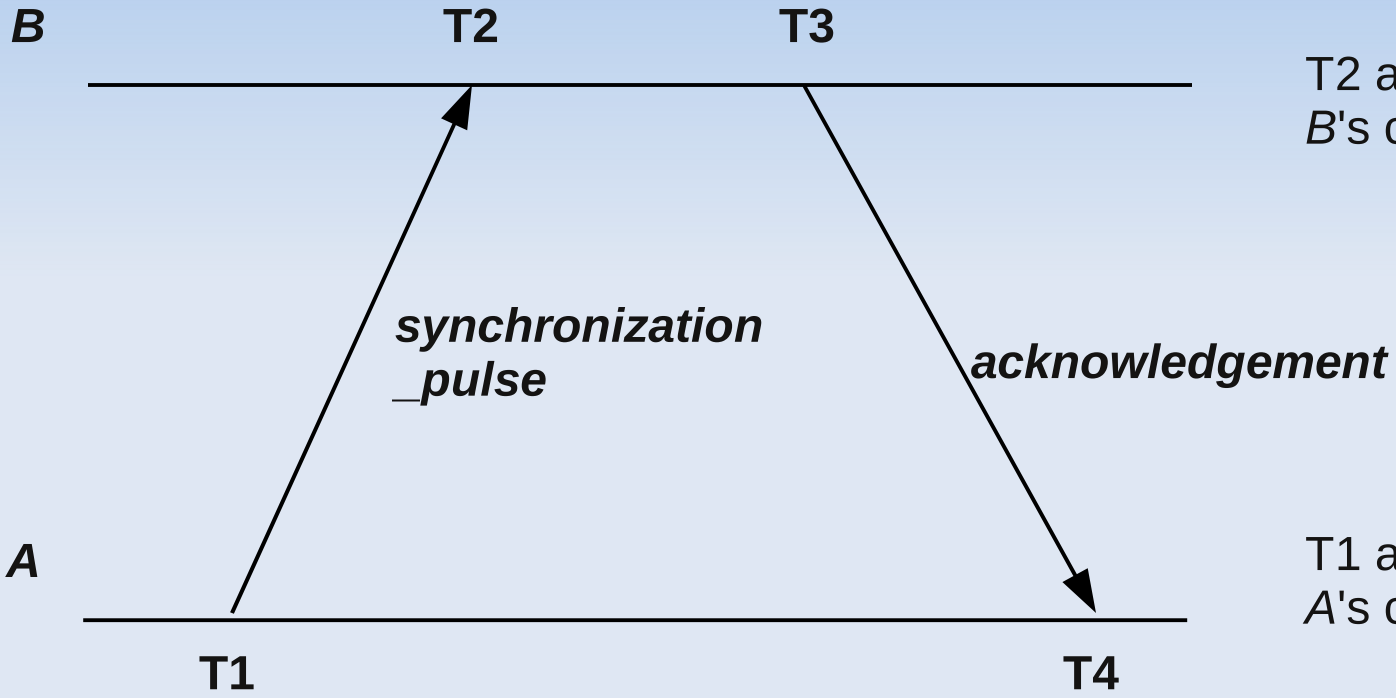
TPSN Protocol

- To provide network-wide time synchronization in a wireless sensor network.
- Two steps
  - Level-Discovering: Establish a hierachical structure in the network.
  - Synchronization: Perform pair wise synchronization along the edges of this structure.
- Result
  - All nodes in the network synchronize their clock with a reference node (the root).

Level-Discovering Phase

- Assign the root node level 0.
- The root broadcasts a *level_discovery* packet carrying its identity and level information to its immediate neighbors.
- The immediate neighbors of the root receive this packet and assign themselves a level, one greater than the level they have received.
- Repeat the process until every node in the network is assigned a level.

Synchronization Phase



T2 and T3 are measured in *B*'s clock

T1 and T4 are measured in *A*'s clock

Two way messge exchange between a child (*A*) and its parent (*B*)

Synchronization Phase(2)

- Let Δ be the clock drift between *A* and *B*.
- Let d be the propagation delay.
- Assume Δ and d do not change in a short time span.

-
  ```
  T2 - T1 = d + Δ    (1)
  T4 - T3 = d - Δ    (2)
    ==> Δ = ((T2-T4)-(T1-T3))/2
  ```
- *A* performs clock adjustment: $t = t + \Delta$

The Uppaal Model Checker

- An integrated tool for specification, simulation and verification of real-time systems.
- Input of Uppaal: the XTA (eXtended Timed Automata) format.
- Clocks in Uppaal:
  - Can only be assigned an integer expression
  - Can only be compared with an integer expression or another clock
  - Clocks can not be read
  - Clocks can not advance by an arbitrary amount

Integer Clock

- To make clocks readable: use integer variables
  - `typedef int intclock;`
- To make clocks advance periodically: use a *Unviserval Pulse Generator* (UPG) process to broadcast a *time_pulse* signal to all processes in the system.

The UPG Process

```
broadcast chan time_pulse;
process universal_pulse_generator()
{
   clock t;
   state S {t <= 1};
   init S;
   trans
        S -> S { guard t == 1;
                 sync time_pulse!;
                 assign t = 0; };
}
```

Integer Clock User

- Processes deploying integer clocks.
- For every state and every integer clock, the integer clock user process must specify a transition that responds to the *time_pulse* event (to advance the integer clock).
- Drawbacks: an increase in model complexity, but the code to implement integer clocks is straightforward.

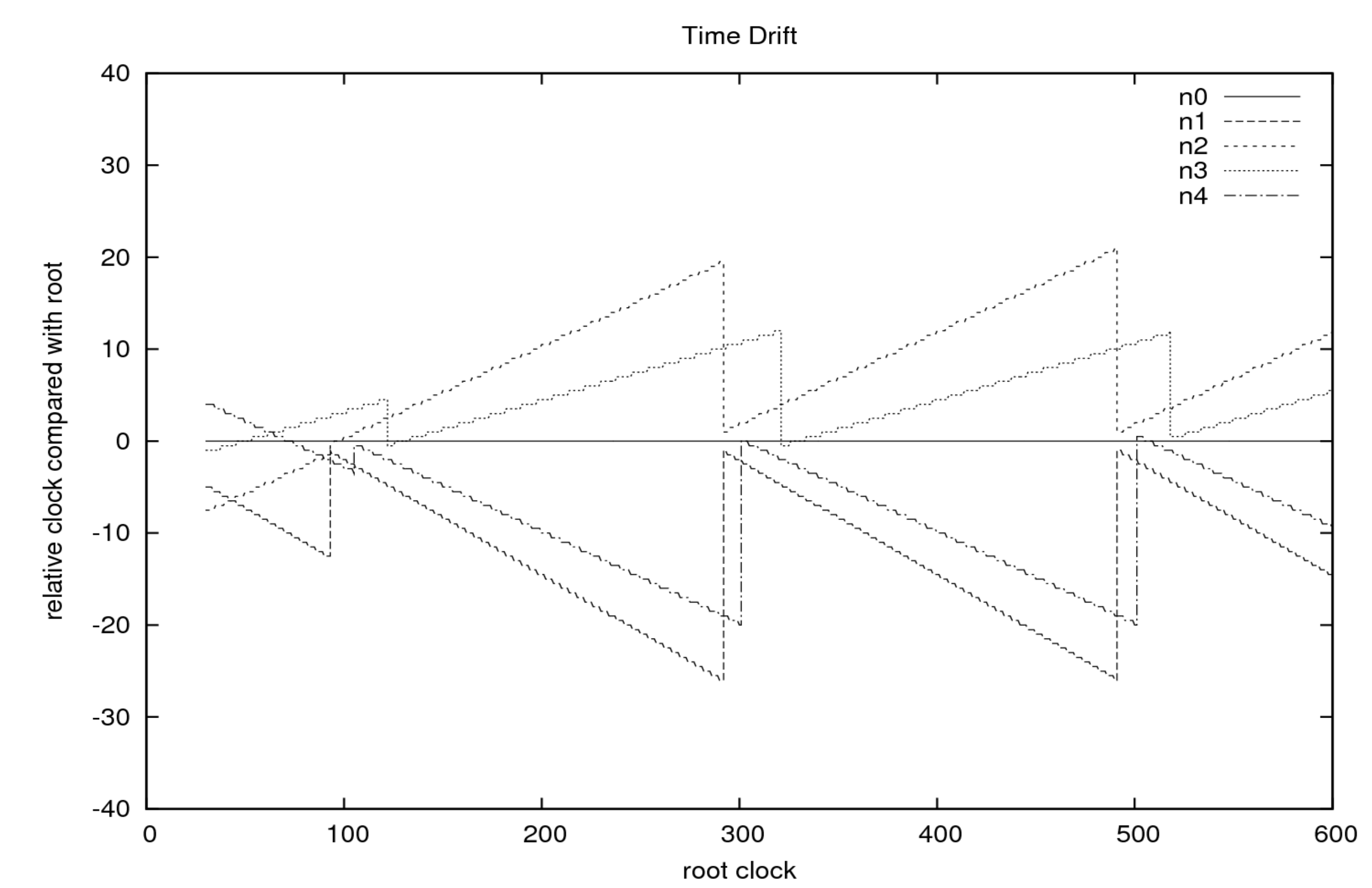
Integer Clock User

```
chan AtoB;
meta int msg;
process A()
{
  const int wait = 3;
  meta intclock x;
  state INIT, SENT;
  init INIT;
  trans
    INIT->INIT { sync time_pulse?; assign x=x+1; },
    SENT->SENT { sync time_pulse?; assign x=x+1; },
    INIT->SENT { guard x >= wait;
                 sync AtoB!; assign msg=x; };
}
```

Modeling TPSN

- Three Uppaal states are used for level-discovery phase and four are used for synchronization phase.
- Additional features
  - Time Drift and Resynchronization: node's clock has drifts from ideal clock. The whole network need to be resynchronized periodically before nodes' clocks diverge too much from the root.
  - Node dying/reviving.

Simulation Result



Verification Result

- No Deadlock:
  A[] not deadlock
- Synchronized:
  A[](<>ni.state == synchronized)
- Relative Time Bounded:
  A[] abs(n_i.local_clock–n_0.local_clock)< X
- Relative Time Close:
  A[](<>abs(n_i.local_clock– n_0.local_clock)<Y)

Verification Result (2)

Size of Network	No Deadlock	Synchronized	Relative Time Bounded	Relative Time Close
3	0.61sec / 21MB	2.06 sec / 24 MB	0.62 sec / 21 MB	2.11 sec / 24 MB
4	6.5 sec / 22MB	68.0 sec / 31 MB	6.7 sec / 22 MB	70.2 sec / 31 MB
5	6.1 min / 126 MB	214.9 min / 181 MB	6.3 sec / 126 MB	236.4 min / 181 MB

Thank You!