Real-time Stream Monitoring with STREAMLAB

Peter Faymonville*, Bernd Finkbeiner*, Malte Schledjewski*, Maximilian Schwenger*,
Leander Tentrup*, Hazem Torfah*
*Reactive Systems Group
Saarland University
Saarbrücken, Germany
lastname@react.uni-saarland.de

Abstract

The online evaluation of real-time data streams is an essential part of cyber-physical systems and plays a crucial role when monitoring those that are intrinsically safety-critical. We present STREAMLAB, a specification and monitoring framework for stream-based real-time properties, which has already been applied on autonomous aerial vehicles. As a stream-based monitoring engine, STREAMLAB translates input streams, for example sensor readings or other data collected at runtime, into output streams containing aggregated statistics. In practice, input data often arrives in varying, unpredictable frequencies and a monitor has to cope with limited memory and computational resources. STREAMLAB tackles these problems by combining the benefits of a programming language and a formal, temporal logic. While it allows for expressing a vast variety of real-time properties, specifications are also compositional and easy to reuse by design, and STREAMLAB can compute guaranteed upper bounds on the memory requirements. A monitor for a given specification will respect these bounds and thus never disrupt the normal operation of a system, notwithstanding the frequency of arriving input values. This makes STREAMLAB a suitable choice for embedded devices with strictly limited memory.

During the execution of a cyber-physical system, a variety of data is collected, ranging from sensor readings to external commands. While some of these input values arrive in a relatively fixed frequency, e.g. a sensor provides a new reading approximately every $200\,\mathrm{ms}$, some inputs arrive entirely unpredictably, especially when taking user behavior into account.

A stream processing engine considers data from each individual input source as its own respective *input stream*. The engine aggregates this data to compute statistical and diagnostic information in *output streams*. When output data violates a specified assertion, an alarm is raised to inform about a potential problem.

A key advantage of this approach is the modular nature of streams, which allows for compositional, easy to reuse specifications. Existing stream-based languages like LOLA [1, 2] are based on the synchronous programming paradigm [3, 4], i.e., they assume the existence of a discrete global clock. In each tick of the global clock, all input streams receive a new sensor value and each output stream is extended by a new computed value. In real systems, this is not necessarily true: consider, for example, an autonomous drone with several sensors such as a GPS module, an inertial measurement unit, and a laser distance meter. While a synchronous arrival of all measured values would be desirable, sensors do not necessarily operate on a common clock so their frequencies can drift apart over time. Moreover, not all measurement frequencies are equal, nor are sensors always reliable. Thus, we consider such a system asynchronous. Monitoring asynchronous systems introduces new challenges and possibilities: in the synchronous settings, for example, it is not possible to detect deviations in the expected arrival frequency, whereas in the asynchronous setting, it is.

We propose using STREAMLAB for monitoring asynchronous systems, which has already been applied successfully to monitor unmanned aerial vehicles in cooperation with the German Aerospace Center [5, 6]. The core of STREAMLAB is the specification language RTLOLA based on the stream-based runtime verification language LOLA. An RTLOLA specification consists of a set of input stream declarations, and several output stream definitions where each stream is equipped with an expression declaring how an output value ought to be computed. Output streams are either *event-based* or *periodic*. Event-based streams are the output counterpart of input streams: new values are generated asynchronously when all inputs relevant to this stream occur in an event. In contrast to that, periodic streams are decoupled from the arrival of input streams and *isochronous*, i.e., they are extended at pre-defined

This work was partially supported by the European Research Council (ERC) Grant OSARES (No. 683300) and by the DFG as part of TRR 248 (No. 389792660)

```
input  TAS, pitch, lon, lat: Float64
output gnd_spd := cos(pitch) * TAS
output tas_dist @5Hz := gnd_spd[∫, 5s]
output gps_spd := sqrt(delta(lon) ^ 2 + delta(lat) ^ 2)
output gps_dist @5Hz := gps_spd[∫, 5s]
output error := gps_dist - tas_dist
trigger error[∑, 10s] > 5 "Sensor values do not match."
```

Figure 1: A RTLOLA specification comparing the estimated distance covered according to the true air speed (TAS), and the GPS values provided by two individual modules. When the accumulated deviation over a 10 s interval exceeds 5 m, an alarm is raised.

points in time. Accesses from event to periodic streams and vice versa are resolved with a zero-order hold. This split brings forth two advantages: periodic streams allow for resource-intensive computations without overloading the monitoring process by accident, and event-based streams allow the monitor to react to incoming values in a timely manner.

Example: Consider the monitor of a fixed-wing aircraft. It gets sensor values from two components. The first one provides the true air speed (TAS) and pitch of the aircraft, whereas the second one is a GPS module providing the current longitude and latitude.

The RTLOLA specification in Figure 1 detects a large accumulated error in the estimated position based on GPS and TAS (the specification is simplified for the sake of illustration and does not take factors like roll or wind into account). For this, output stream `gnd_spd` computes the ground speed using TAS and the current pitch whenever a new sensor reading for both sensors arrive. The `tas_dist` stream computes the distance the aircraft traveled within 5 seconds according to the sensed TAS by integrating the ground speed. The integration is a *sliding window*, i.e., whenever `tas_dist` is computed, it uses all values of `gnd_spd` occurring in the last 5 seconds. This decouples the frequencies of `tas_dist` and `gnd_spd`. To get the velocity according to the GPS location in output `gps_spd`, we use Pythagoras' theorem with the difference in two consecutive GPS readings, denoted by `delta(lon)` and `delta(lat)`. Output stream `gps_dist` then computes the sliding integral over 5 seconds with a decoupled frequency of 5 Hz. The stream `error` computes the actual deviation between the two measured distances. Lastly, **`trigger`** defines a threshold for the absolute derivation of both distances and raises an alarm if they exceed 5 m. STREAMLAB infers that the trigger and the `error` output stream need to be computed with frequency 5 Hz because all streams they depend on use this frequency.

RTLOLA simplifies the specification process by providing directives for complex constructs such as the integral over a sliding window of values. Moreover, extending the specification to detect e.g. hover periods is straightforward: simply add **`output`** `hover @10Hz := gnd_spd[∫, 1s] < 0.2` and a suitable trigger. The strict decoupling of periodic and event-based streams allows the specifier to have fine-grained control over when expensive computations take place, and thus reduces stress on the monitor component in terms of CPU utilization. Since STREAMLAB analyzes the specification and determines an upper bound on the memory consumption, the user can verify an undisturbed monitoring process ahead of deployment.

REFERENCES

[1] B. D'Angelo, S. Sankaranarayanan, C. Sánchez, W. Robinson, B. Finkbeiner, H. B. Sipma, S. Mehrotra, and Z. Manna, "LOLA: runtime monitoring of synchronous systems," in *12th International Symposium on Temporal Representation and Reasoning (TIME 2005), 23-25 June 2005, Burlington, Vermont, USA*, 2005, pp. 166–174. [Online]. Available: https://doi.org/10.1109/TIME.2005.26

[2] P. Faymonville, B. Finkbeiner, S. Schirmer, and H. Torfah, "A stream-based specification language for network monitoring," in *Runtime Verification - 16th International Conference, RV 2016, Madrid, Spain, September 23-30, 2016, Proceedings*, 2016, pp. 152–168. [Online]. Available: https://doi.org/10.1007/978-3-319-46982-9_10

[3] G. Berry and G. Gonthier, "The esterel synchronous programming language: Design, semantics, implementation," 1992.

[4] N. Halbwachs, F. Lagnier, and C. Ratel, "Programming and verifying real-time systems by means of the synchronous data-flow language LUSTRE," *IEEE Trans. Software Eng.*, vol. 18, no. 9, pp. 785–793, 1992. [Online]. Available: https://doi.org/10.1109/32.159839

[5] F. Adolf, P. Faymonville, B. Finkbeiner, S. Schirmer, and C. Torens, "Stream runtime monitoring on UAS," *CoRR*, vol. abs/1804.04487, 2018. [Online]. Available: http://arxiv.org/abs/1804.04487

[6] ——, "Stream runtime monitoring on UAS," in *Runtime Verification - 17th International Conference, RV 2017, Seattle, WA, USA, September 13-16, 2017, Proceedings*, 2017, pp. 33–49. [Online]. Available: https://doi.org/10.1007/978-3-319-67531-2_3