Rivulet: A Fault-Tolerant Platform for Smart-Home Applications

Masoud Saeida Ardekani

Rayman Preet Singh

Nitin Agrawal

Douglas B. Terry

Riza O. Suminto

Middlware 2017

Rivulet

- Fault tolerant distributed platform for smart home application
  - Link loss, network partitions, sensor failures, device crashes

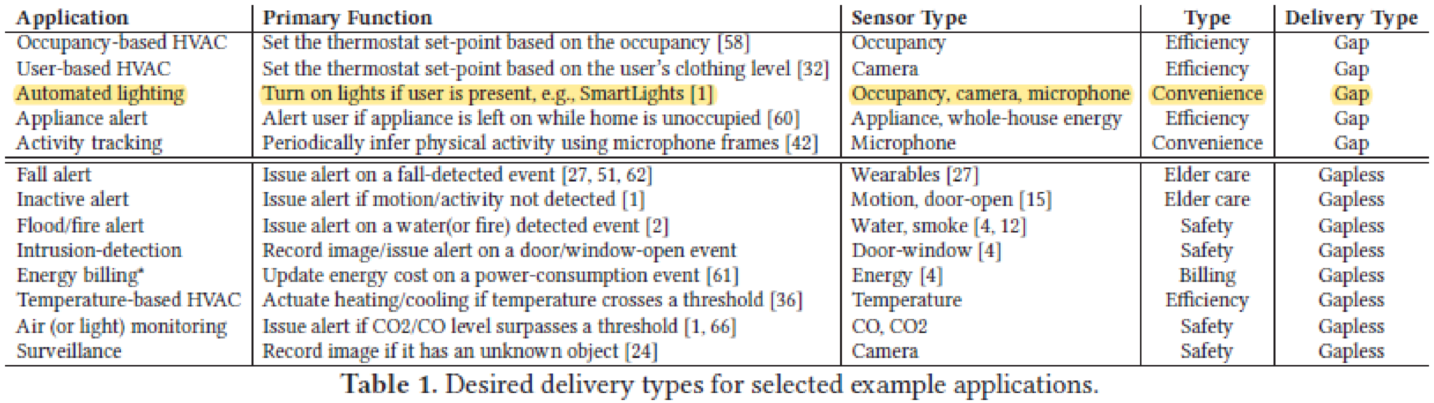
- Previous systems are cloud-centric
  - Home hub communicates events to cloud where apps run, events flow across the WAN
  - Slow, failure-prone

- Rivulet is home-centric
  - Execute everything in the home

Model

- Hub and/or local processing devices
-
- Sensors/Actuators
  - Motion sensors, doors
  - Sensors generate event streams
- Problem: fault tolerance
  - Reliable communication with sensors, skew
  - Process failures (cloud has much stronger guarantees)
  - Gaps in event stream (intrusion, elderly person, …)

Communication Demands



Application	Primary Function	Sensor Type	Type	Delivery Type
Occupancy-based HVAC	Set the thermostat set-point based on the occupancy [58]	Occupancy	Efficiency	Gap
User-based HVAC	Set the thermostat set-point based on the user's clothing level [32]	Camera	Efficiency	Gap
Automated lighting	Turn on lights if user is present, e.g., SmartLights [1]	Occupancy, camera, microphone	Convenience	Gap
Appliance alert	Alert user if appliance is left on while home is unoccupied [60]	Appliance, whole-house energy	Efficiency	Gap
Activity tracking	Periodically infer physical activity using microphone frames [42]	Microphone	Convenience	Gap
Fall alert	Issue alert on a fall-detected event [27, 51, 62]	Wearables [27]	Elder care	Gapless
Inactive alert	Issue alert if motion/activity not detected [1]	Motion, door-open [15]	Elder care	Gapless
Flood/fire alert	Issue alert on a water(or fire) detected event [2]	Water, smoke [4, 12]	Safety	Gapless
Intrusion-detection	Record image/issue alert on a door/window-open event	Door-window [4]	Safety	Gapless
Energy billing*	Update energy cost on a power-consumption event [61]	Energy [4]	Billing	Gapless
Temperature-based HVAC	Actuate heating/cooling if temperature crosses a threshold [36]	Temperature	Efficiency	Gapless
Air (or light) monitoring	Issue alert if CO2/CO level surpasses a threshold [1, 66]	CO, CO2	Safety	Gapless
Surveillance	Record image if it has an unknown object [24]	Camera	Safety	Gapless

Table 1. Desired delivery types for selected example applications.

- Gap: can tolerate drops
- Gapless: cannot

Challenges

- Home is not a data center
  - No central admin
  - Limited redundancy
  - Unique failure modes: plugs, physical interference, battery, up to 14% downtime

- Diverse wireless networks

Rivulet Design

- Rivulet is a local process, runs on: hub, phone, tablet, some appliances
  - Event delivery, execution service
- Rivulets communicate to each other via home wifi
- Failed processes eventually recover

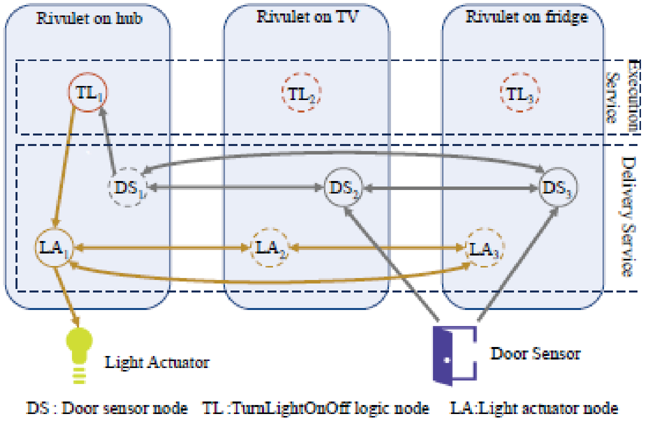
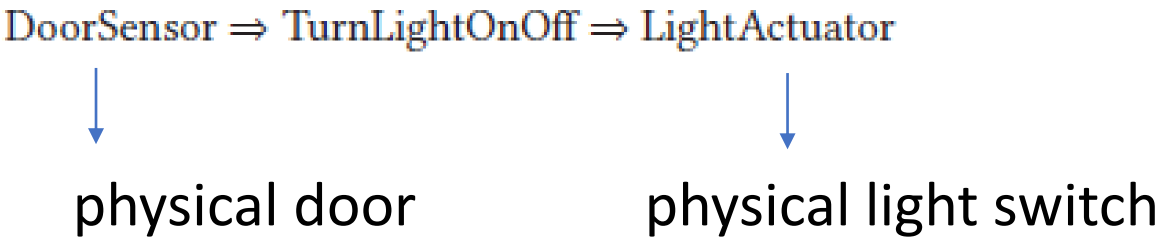


Figure 2. Rivulet System

- Sensor crash: no value returns, eventually reboots
- Actuator crash: does not respond to events, eventually reboots
- Sensors/actuators can may communicate to multiple processes

Rivulet Apps

- DAG
  - Sensors, logic, actuators



DoorSensor ⇒ TurnLightOnOff ⇒ LightActuator

physical door

physical light switch

Inside a Rivulet

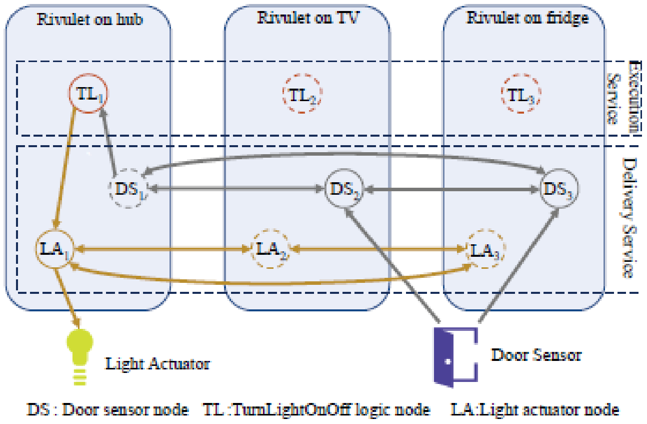


Figure 2. Rivulet System

Each process creates:

active node: (solid) if can communicate directly

shadow node (dashed) otherwise

Action:

event must be received by active node

Computation:

logic node (solid) performs computation

shadow node (dashed) inactive can activate on process failure

Delivery Service

- Push (“`door is open`” event) and pull-based sensors (“`get temp`” event)
- Event ingest component: fetches sensor events, delivers actuator commands
- Event forwarding component: forwards events to logic nodes

- Gapless: polling based, post-ingest (an event is received by one process)
  - Coordinated epoch-based polling; avoid extraneous sensor requests, forward sensor values
  - Event forwarding: replicate ingested event at ALL processes

Gap{less} protocol

- Gapless: ring-based (gossip) between processes
  - Forward to your reachable neighbors, and so on, … suppress dups
  - Fall back to broadcast
  - Stronger failure guarantees
- Gap
  - Only one active node will poll a given sensor
  - If that processes fails, in next epoch, another active node (process) is chosen
  - Limited chain communication: e.g. hub, tv, fridge

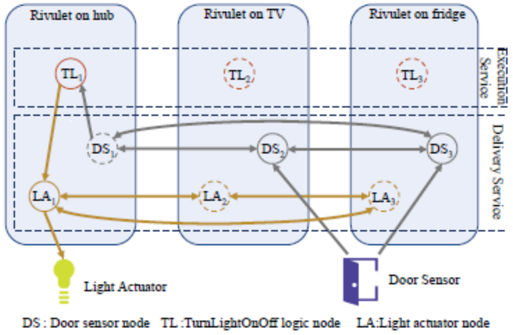


Figure 2. Rivulet System

Application Fault Tolerance

- Primary/second approach for active logic node
- Care must be taken for non-idempotent actions:

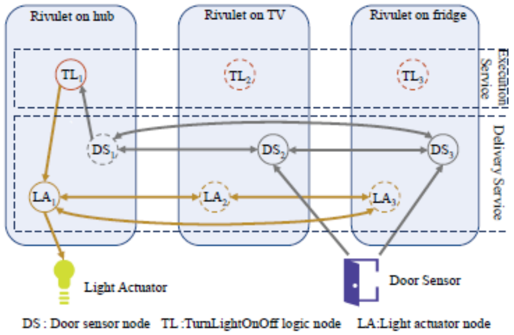
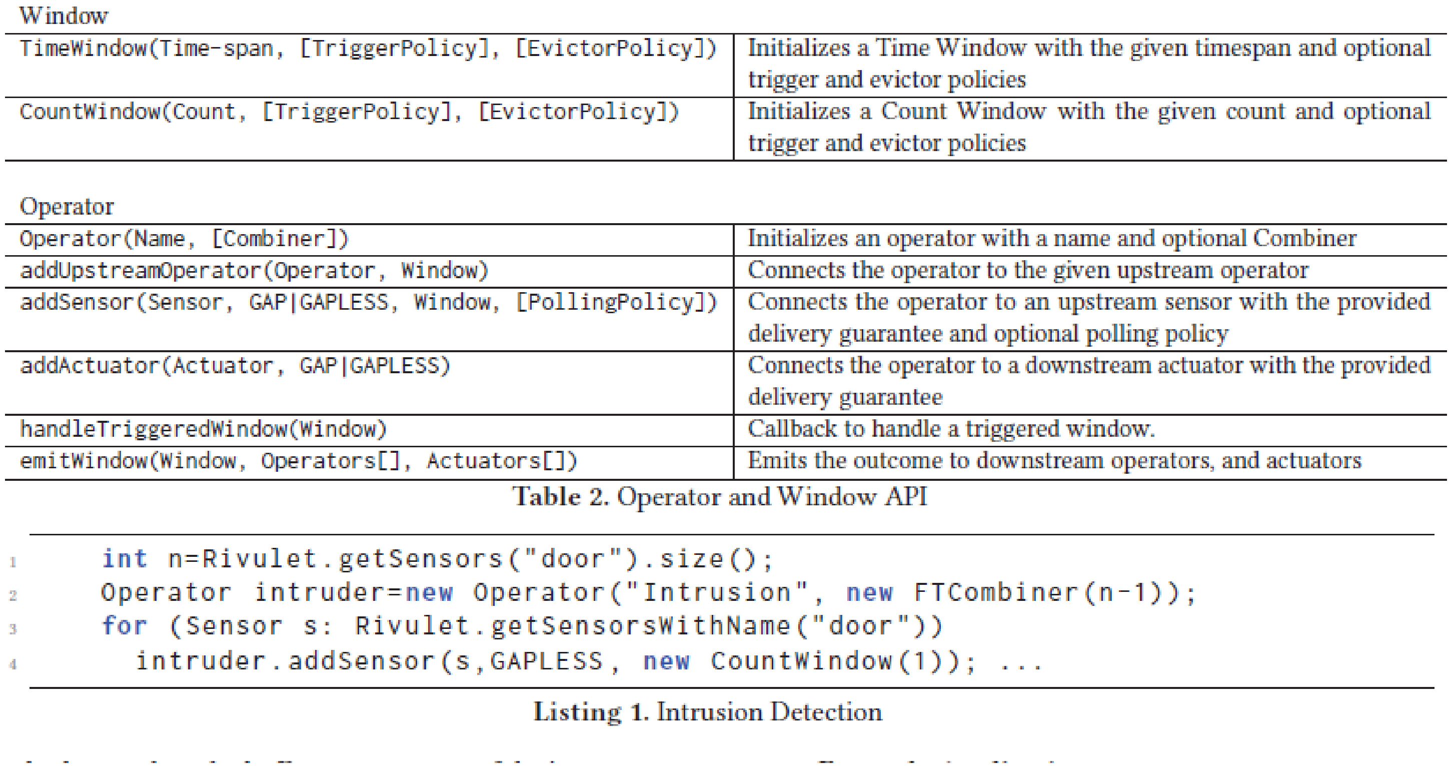


Figure 2. Rivulet System

Programming Model

- DAG model
- Event model: time window, trigger to deliver them, evictor to purge them



Window	
`TimeWindow(Time-span, [TriggerPolicy], [EvictorPolicy])`	Initializes a Time Window with the given timespan and optional trigger and evictor policies
`CountWindow(Count, [TriggerPolicy], [EvictorPolicy])`	Initializes a Count Window with the given count and optional trigger and evictor policies

Operator	
`Operator(Name, [Combiner])`	Initializes an operator with a name and optional Combiner
`addUpstreamOperator(Operator, Window)`	Connects the operator to the given upstream operator
`addSensor(Sensor, GAP|GAPLESS, Window, [PollingPolicy])`	Connects the operator to an upstream sensor with the provided delivery guarantee and optional polling policy
`addActuator(Actuator, GAP|GAPLESS)`	Connects the operator to a downstream actuator with the provided delivery guarantee
`handleTriggeredWindow(Window)`	Callback to handle a triggered window.
`emitWindow(Window, Operators[], Actuators[])`	Emits the outcome to downstream operators, and actuators

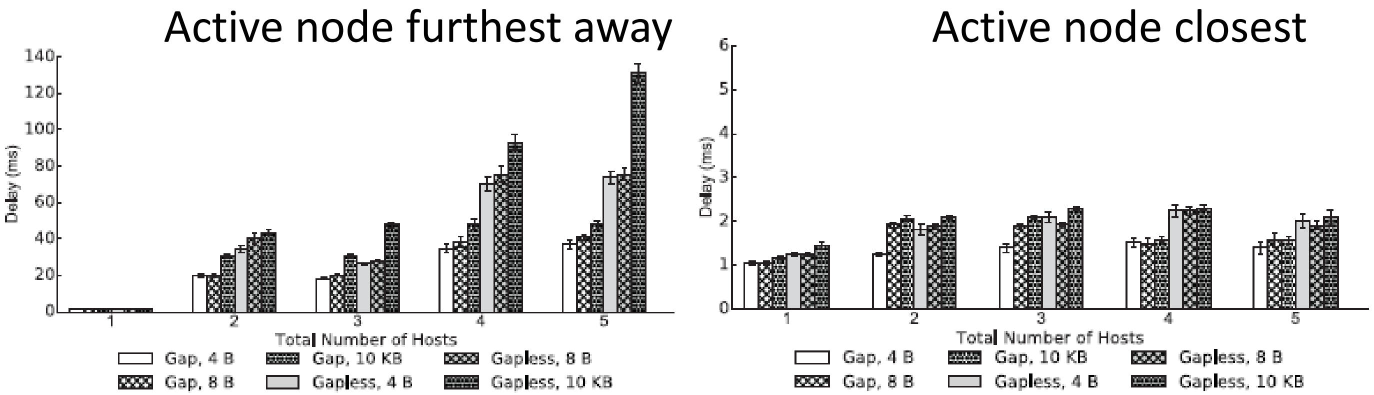
Table 2. Operator and Window API

```
int n=Rivulet.getSensors("door").size();
Operator intruder=new Operator("Intrusion", new FTCombiner(n-1));
for (Sensor s: Rivulet.getSensorsWithName("door"))
  intruder.addSensor(s,GAPLESS, new CountWindow(1)); ...
```

Listing 1. Intrusion Detection

Evaluation: performance

- Java prototype + raspberry pi's around the home, software sensor
- Delay: time between event emitted by a sensor -> active logic node



Active node furthest away

Active node closest

(a) *Worst* topology.

(b) *Best* topology.

Figure 4. Delay incurred with increasing number of processes, for different event sizes.

Evaluation: faults

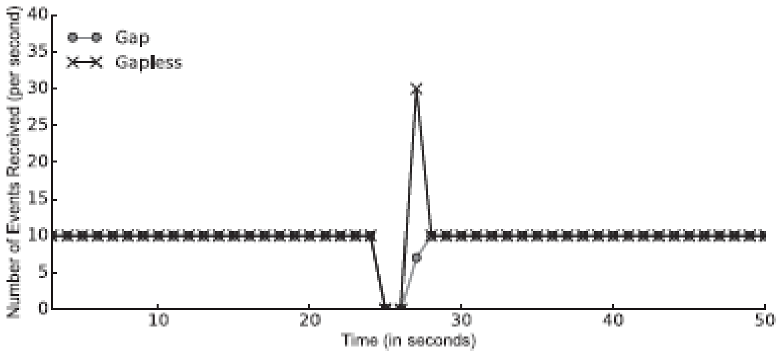


Figure 7. Number of events received by an active logic node. Induced process failure at $t = 24$ seconds.

Discussion