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

Masoud Saeida Ardekani
Rayman Preet Singh
Nitin Agrawal
Douglas B. Terry
Riza O. Suminto

Middleware 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

<table>
<thead>
<tr>
<th>Application</th>
<th>Primary Function</th>
<th>Sensor Type</th>
<th>Type</th>
<th>Delivery Type</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Occupancy based HVAC</td>
<td>Set the thermostat set point based on the occupancy [58]</td>
<td>Occupancy</td>
<td>Efficiency</td>
<td>Gap</td>
<td></td>
</tr>
<tr>
<td>User-based HVAC</td>
<td>Set the thermostat set point based on the user’s clothing level [32]</td>
<td>Camera</td>
<td>Efficiency</td>
<td>Gap</td>
<td></td>
</tr>
<tr>
<td>Automated lighting</td>
<td>Turn on lights if user is present, e.g., SmartLights [1]</td>
<td>Occupancy, camera, microphone</td>
<td>Convenience</td>
<td>Gap</td>
<td></td>
</tr>
<tr>
<td>Appliance alert</td>
<td>Alert user if appliance is left on while home is unoccupied [60]</td>
<td>Appliance, whole-house energy</td>
<td>Efficiency</td>
<td>Gap</td>
<td></td>
</tr>
<tr>
<td>Activity tracking</td>
<td>Periodically infer physical activity using microphone frames [42]</td>
<td>Microphone</td>
<td>Convenience</td>
<td>Gap</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Full alert</th>
<th>Inactive alert</th>
<th>Flood/fire alert</th>
<th>Intrusion detection</th>
<th>Temperature-based HVAC</th>
<th>Air (or light) monitoring</th>
<th>Surveillance</th>
</tr>
</thead>
</table>

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

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

- DAG
  - Sensors, logic, actuators

\[
\text{DoorSensor} \Rightarrow \text{TurnLightOnOff} \Rightarrow \text{LightActuator}
\]

- physical door
- physical light switch
Inside a Rivulet

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
Application Fault Tolerance

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

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

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

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