Home, SafeHome: Ensuring a Safe and Reliable Home Using the Edge

Shegufta Ahsan*, Rui Yang*, Shadi Noghabi**, Indranil (Indy) Gupta*

Usenix HotEdge 2019

http://dprg.cs.uiuc.edu/ :: http://indy.cs.illinois.edu
(* Univ. of Illinois Urbana-Champaign, ** Microsoft Research)
1. How many of you have IoT devices in your (smart) home?

2. How many of you use the same app (on your mobile device) to control MORE than 1 of the IoT devices in your smart home?

3. How many of you use the same app to control ALL the IoT devices in your smart home?
Smart Homes

• “All media are extensions of some human faculty -- psychic or physical.”
  -- Marshall McLuhan.

• Not true in smart homes/buildings today!

1. Users today control smart homes and buildings in a largely manual style.
   • Users directly control devices, e.g., via mobile
   • Imperative programming (e.g., Routine = Sequence of Commands) comes with correctness issues

2. Additionally, Humans today manually ensure that safety properties are not violated.
   • Stove is ON => Exhaust fan is ON
   • House LOCKED => Security cameras ON
   • ATMOST (1) (South Lawn Sprinklers, North Lawn Sprinklers)
The State Today

- **Routines** (sequences of commands) that are concurrent can conflict with each other, creating **inconsistent outcomes** and **unsafe states**
  - Humans cannot reason about concurrency at millisecond-level
- **Erroneous routines** may violate Safety Properties
  - Switch OFF Exhaust Fan; Switch ON Stove;
- **Failures of devices** have unintended consequences and result in inconsistent outcomes and unsafe states
Two Concurrent Routines

R2 starts soon after R1
→ More final states are inconsistent

Worse with longer routines

SafeHome’s goal:
Bring all lines down to horizontal axis

Routines R1 and R2 run on (X-axis) TP-Link HS105 smart plugs.
R1 turns on all lights, then R2 turns off all lights.
Times above (ms) show time gap from R1 start to R2 start.
SafeHome

• A software-defined management approach for smart home management.

• Features:
  1. Users specify home-wide Safety properties in a declarative way – SafeHome ensures these all the time (disallows or aborts routines that violate)
  2. Users can imperatively program routines
  3. SafeHome Autonomously catches and responds to concurrency conflicts, safety violations, and failures.
  4. Modular design
  5. Sits on edge, and works with commodity devices and APIs (no modifications of device)
  6. Avoids putting logic on cloud, which would have increased latency and violated privacy
ASID Challenges

• **A:** SafeHome-Atomicity. Execution of a routine is atomic and exactly-once.
  • When a routine finishes, either: a) all its commands have been executed successfully, or b) none of its commands have had an effect on the smart home.
  • **Challenges:** a) catching conflicts, b) aborting routines, c) undo-ing routines.

• **S:** SafeHome-Safety. User-specific Safety properties are satisfied at runtime.
  • **Challenges:** a) Safety properties span multiple devices, b) catching these at run-time.

• **I:** SafeHome-Isolation. Concurrent routines are isolated from interfering with each other at devices.
  • **Challenges:** If routines interfere, SafeHome must ensure the execution is serially equivalent.

• **D:** SafeHome-Durability. A routine that completes successfully cannot be undone (except by another subsequent routine).
  • **Challenges:** No undo after commit point of routine.
Safety Properties: SafeHome’s Grammar

A:- if A then A else A
A:- DeviceID.StateID ==<>!= foo
A:- ALL(A), ANY(A), !A, ATLEAST(k)(A), ATMOST(k)(A), A AND A, A OR A

This is a first-cut grammar. Surprisingly, captures a wide swathe of safety specifications.
## Safety Specifications: Examples

<table>
<thead>
<tr>
<th>Undesirable State</th>
<th>Desirable Safety Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routine R1 turns on both stove and exhaust-fan, but then Routine R2 turns off exhaust-fan.</td>
<td>IF (stove==ON) THEN (exhaust-fan==ON)</td>
</tr>
<tr>
<td>Routine R1 opens a window, Routine R2 turns on air-conditioner.</td>
<td>IF (air-cond==ON) THEN (windows==CLOSED)</td>
</tr>
<tr>
<td>Power overload due to multiple heavy devices.</td>
<td>IF (dishwasher==on) THEN ATMOST(1) (washingmachine==ON, dryer==ON)</td>
</tr>
<tr>
<td>Turning on all sprinklers around the house leads to insufficient water pressure.</td>
<td>ATMOST(1) (Northeast-sprinkler==ON, Northwest-sprinkler==ON, Southeast-sprinkler==ON)</td>
</tr>
<tr>
<td>User accidentally leaves garage-door open overnight.</td>
<td>IF (garage-door.OPEN &gt; `n' hours) THEN (garage-door==CLOSE)</td>
</tr>
</tbody>
</table>
Failures and Safety

• Safety properties are impossible to guarantee always
  • Stove and Exhaust fan are both ON → Exhaust fan fails

• SafeHome ensures safety properties are invalid for at most a tolerance window (after a failure)
  • Could be set by user or physical constraints (e.g., reboot time)

• SafeHome uses tolerance window to set timeout in its failure detector algorithm
Where it **Really** Gets Interesting (1/2)

I. ASID@IoT Mechanisms can borrow heavily from ACID@Database mechanisms. But key differences:

• ASID@IoT optimizes **latency** and abort rate, while ACID@DB optimizes **throughput** and abort rate.

• **Intermediate states** in ASID@IoT are almost always visible to user (may not be in ACID)
  • Undo of routine needs to have consolidated action across affected devices

• **Long-running routines** exist in ASID@IoT (rarer in ACID)
  • Run North Sprinklers for 15 minutes; Run South Sprinklers for 20 minutes;
  • Challenges: a) Interaction between long-running and short-running (instant) routines; b) Interaction among long-running routines.

• **Human Interrupts, Exceptions, Pauses**

• **Concurrent Control: Optimistic vs. Pessimistic Approaches**
Where it **Really** Gets Interesting (2/2)

II. **Safety Checking** can borrow from Static and Dynamic Type Checking in Compilers/Programming Languages. But:

- Dynamic checking need to deal with a) concurrent routines, b) failed devices that may or may not recover (optimistic abort vs. pessimistic abort)

III. Interesting **dilemmas**

- Goto Dilemma: Should the default state (after-failure reboot) for garage door be OPEN or CLOSED?
  - OPEN = Hello, Burglars!
  - CLOSED = Door closes on a car underneath it.
- Also occur in self-driving cars (Tesla Model S fatality May 2016, Ohio)
Feedback/Controversial/Open Qs/Fall Apart

• Latency
  • Biggest need, and main reason for system to fall apart: “it’s too slow!”
  • DB ACID consistency literature: useful? How deep? (our focus: Latency)

• User involvement
  • UI: Need an easy UI for specifying safety properties, and for programming routines.
  • Is ASID behavior (esp. abort and undo) cumbersome to user?
  • Cannot (always) require human intervention. E.g., deadlocks, safety violations.

• Device Resources: SafeHome assumes no extra capability or memory on devices. With more capable devices:
  • More capable devices can be used for failure recovery when edge is down, eliminating cloud reliance.
  • Such smart devices can serve as failover for edge device (run SafeHome logic).

• ACID: Downsides?
SafeHome

- A software-defined management approach for smart home management.

Features:
1. Users specify home-wide Safety properties in a declarative way – SafeHome ensures these all the time (disallows or aborts routines that violate)
2. Users can imperatively program routines
3. SafeHome Autonomously catches and responds to concurrency conflicts, safety violations, and failures.
4. Modular design
5. Sits on edge, and works with commodity devices and APIs (no modifications of device)
6. Avoids putting logic on cloud, which would have increased latency and violated privacy
SafeHome Architecture

- Routine Manager
- Concurrency Controller
- Safety Checker
  - Static
  - Dynamic
- Command Deployer
- Definition Bank
- State & Health Tracker
# Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>device</td>
<td>a smart home device with a set of potential states</td>
</tr>
<tr>
<td>command</td>
<td>a user/program triggered instruction that changes the state of an individual device</td>
</tr>
<tr>
<td>routine</td>
<td>a sequence of commands</td>
</tr>
<tr>
<td>Safety properties</td>
<td>guaranteed device behaviors that user expects from the smart home</td>
</tr>
</tbody>
</table>
**People: Needs & Wants**

1. FAT: Fairness, Accountability, Transparency, Bias, Individual/group
2. I.E.: Interpretability, Explainability
3. Democratization: Equality, Equity
4. Education
5. Legal e.g., GDPR, HIPAA
6. Ethics
7. Declarative Programming
8. Security Privacy, Confidentiality, Integrity
9. Reliability
10. Scale & Fault-tolerance

**Future of People**

A. Future of Health
B. Future of Relationships
C. Future of Employment (job finding, task matching, team making)
D. Future of Transportation
E. Future of News
F. Future of Agriculture
G. Future of Communities
H. Future of Markets
I. Future of Peace
J. Future of Programming
K. Future of Research
L. Future of Peace

**Intelligent Infrastructures**

I. Social Media
II. Intelligent Web
III. IoT Smart cities, Smart vehicles, Smart*
IV. Finance
V. Energy Oil, Gas, Nuclear
VI. Utilities
VII. Materials & Manufacturing
VIII. Healthcare
IX. Supply-Chain
X. Internet & Telecom
XI. Datacenters & Clouds
XII. Defense

**Computer Science**

*Systems Researchers Need to do this more!*

*Systems Researchers Do These Very well!*