Runtime Monitoring of Human Behaviour with Aggregate Computing on Android
VORTEX 2023

Volker Stolz†, Giorgio Audrito*

*University of Turin, Italy
†Western Norway University of Applied Sciences, Bergen

@fm_volker@mastodon.social

July 18, 2023
Overview

- Aggregate Programming in the field calculus with FCPP
- A temporal and spatial logics
- Applications & experiments:
  - “evacuation”, “friend-finding”
- Architecture of the Android application framework
  - Bluetooth Low Energy (BLE) “Advertisement” and “Scanning”
  - Reuse of code-base for simulation and deployment
Where can we use Aggregate Computing?

distance estimation, data summarisation (event detection), selecting areas (network partitioning, channel establishment...), inducing shapes (crowd dispersion, formation control...)... and others!
Where can we use Aggregate Computing?

distance estimation, data summarisation (event detection), selecting areas (network partitioning, channel establishment...), inducing shapes (crowd dispersion, formation control...),... and others!
Why are distributed systems hard to deal with?

diverse heterogeneous entities
  - different computing power
  - sensing and actuation capabilities

We need . . .
  - device abstraction
  - multi-platform frameworks
  - not too bad so far . . .
Why are distributed systems hard to deal with?

diverse heterogeneous entities
- different computing power
- sensing and actuation capabilities
Why are distributed systems hard to deal with?

diverse heterogeneous entities

- different computing power  Still kind of true
- sensing and actuation capabilities  ditto
Formal model: Event structures

- a set of events \( E \)
- a DAG of messages \( \leadsto \)
- a causality partial order \( < \) (transitive closure of \( \leadsto \))
A Simple Concrete Computational Model

simplifying assumptions...
- the same program is executed in every event
- ...can still execute different code through branching
- messages are sent through broadcast (can extend to pointwise messages)

Round:
1. gather data received, stored and sensed
2. compute the program
3. broadcast the result to neighbours
4. perform actuation as computed
5. receive messages while sleeping
Principal Coordination Construct: \texttt{nbr(e)}

- represents \textit{interaction} between neighbour devices
- sends result of \texttt{e} to neighbours (\textit{duality} outgoing - incoming)
- collects neighbour's values for the same \texttt{e} into a \textit{neighbouring field}

\[\text{nbr}(\texttt{e}_c)\]

\textit{neighbouring field of counters.}

\[\texttt{e}_c \rightarrow 2, \text{ broadcast 2}\]

\[\text{nbr}(\texttt{e}_c) \rightarrow \phi\text{ where}\]

\[\phi = \delta_2 \rightarrow 1, \delta_3 \rightarrow 2, \delta_4 \rightarrow 3\]
Principal Coordination Construct: \texttt{nbr(e)}

- represents \textit{interaction} between neighbour devices
- sends result of \texttt{e} to neighbours (\textit{duality} outgoing - incoming)
- collects neighbour’s values for the same \texttt{e} into a \textit{neighbouring field}

\begin{equation}
\text{sum\_hood(nbr(1))}
\end{equation}

counts the number of neighbours

Other functions on fields:
- \texttt{sum\_hood}
- \texttt{min\_hood}
- \texttt{all\_hood}
- \texttt{any\_hood}
Syntax & Semantics

Syntax of past-CTL and SLCS

\[ \phi ::= \bot \mid \top \mid q \mid (\neg \phi) \mid (\phi \land \phi) \mid (\phi \lor \phi) \mid (\phi \Rightarrow \phi) \mid (\phi \Leftrightarrow \phi) \mid (P \phi) \mid (AP \phi) \mid (EP \phi) \mid (H \phi) \mid (AH \phi) \mid (EH \phi) \mid (Y \phi) \mid (AY \phi) \mid (EY \phi) \mid (\phi S \psi) \mid (\phi AS \phi) \mid (\phi ES \phi) \mid (\Box \phi) \mid (\Diamond \phi) \mid (\partial \phi) \mid (\partial^- \phi) \mid (\partial^+ \phi) \mid (\phi R \phi) \mid (\phi T \phi) \mid (\phi U \phi) \mid (G \phi) \mid (F \phi) \]

Temporal & spatial scope:

- \( Y \phi \): “\( \phi \) held in the previous event on the same device”;
- \( EY \phi \): “\( \phi \) held in some previous event on any device”;
- \( \phi S \psi \): “\( \psi \) held in some past event on the same device, and \( \phi \) has held on the same device since then”;
- \( \phi AS \psi \) (resp. \( \phi ES \psi \)): “for all paths (resp. exists a path) of messages reaching the current event, \( \psi \) held in some event of the path and \( \phi \) has held since then”.

Syntax & Semantics

Syntax of past-CTL and SLCS

\[ \phi ::= \bot \mid \top \mid q \mid (\neg \phi) \mid (\phi \land \phi) \mid (\phi \lor \phi) \mid (\phi \Rightarrow \phi) \mid (\phi \Leftrightarrow \phi) \quad \text{logical} \]
\[ \mid (P \phi) \mid (AP \phi) \mid (EP \phi) \mid (H \phi) \mid (AH \phi) \mid (EH \phi) \quad \text{temporal} \]
\[ \mid (Y \phi) \mid (AY \phi) \mid (EY \phi) \mid (\phi S \phi) \mid (\phi AS \phi) \mid (\phi ES \phi) \]
\[ \mid (\Box \phi) \mid (\Diamond \phi) \mid (\partial \phi) \mid (\partial^- \phi) \mid (\partial^+ \phi) \quad \text{spatial} \]
\[ \mid (\phi R \phi) \mid (\phi T \phi) \mid (\phi U \phi) \mid (G \phi) \mid (F \phi) \]

Temporal & spatial scope:

- \( \Box \phi \) (interior): true at points where all neighbours satisfy \( \phi \);
- \( \Diamond \phi \) (closure): true at points where a neighbour satisfies \( \phi \);
- \( \partial, \partial^- \) and \( \partial^+ \): boundary (closure without interior), interior boundary (set without the interior) and closure boundary (closure without the set).
Runtime Monitors in FCPP

```cpp
androidDemoApp/fcpp-android/lib/coordination/past_ctl.hpp

46  /// @brief f1 holds since f2 held in the same device.
47  FUN bool S(ARGS, bool f1, bool f2) { CODE
48       return old(CALL, false, [&](bool o) -> bool {
49           return f2 | (f1 & o);
50       });
51  }
52
53  /// @brief f1 holds since f2 held in all devices.
54  FUN bool AS(ARGS, bool f1, bool f2) { CODE
55       return nbr(CALL, false, [&](field<bool> n) -> bool {
56           return f2 | (f1 & all_hood(CALL, n));
57       });
58  }
59
60  /// @brief f1 holds since f2 held in any device.
61  FUN bool ES(ARGS, bool f1, bool f2) { CODE
62       return nbr(CALL, false, [&](field<bool> n) -> bool {
63           return f2 | (f1 & any_hood(CALL, n));
64       });
65  }
```
Evacuation Experiment

- App partitions user into “left” or “right” group (randomly)
- On evacuation-begin, timer starts (manually)
- Phone-display shows group-membership
- Subjects evacuate according to their group
- Expected outcome: App shows groups eventually correctly partitioned (or “traitor” detected)
Some results:

- Works “well” with sub-second period.
- Visible load on battery.
- Additional “friend-finding” experiment (a la “hot & cold”) more challenging (flakiness, low N, UI/instruction issue)
Evacuation Experiment: Properties

- \( ED \): “Evacuation Done”; time-limit reached.
- \( L \): user is part of the “left” group, false otherwise.
- \( \phi_{HG} = (L \Rightarrow G L) \land (\neg L \Rightarrow G \neg L) \): user is part of homogeneous group.
- \( \phi_{TF} = AH(ED \Rightarrow \phi_{HG}) \): “traitors” found at end of experiment.

Operators:

- \( G \phi, F \phi \) (everywhere, somewhere): true where \( \phi \) holds in every (resp. some) point of every (resp. some) incoming path. Here: “If the user is part of the left group, then everyone in its connected area should also be in the left group; and similarly for the right group.”
- \( AH(ED \Rightarrow \phi_{HG}) \): “it has always and everywhere been the case that after the evacuation is done everyone is within an homogeneous group”
App Architecture

FCPP main features — https://github.com/fcpp/

- **C++ library** used to develop distributed programs using it
  - manipulates C/C++ values
  - can use external C/C++ code
  - portable to any architecture with C++ compiler
- extensible component-based architecture
- runtime monitors for spatio-temporal properties on top of FCPP primitives
App Architecture

FCPP main features — https://github.com/fcpp/

- **C++ library** used to develop distributed programs using it
  - manipulates C/C++ values
  - can use external C/C++ code
  - portable to any architecture with C++ compiler

- Here: cross-compiled to Android architectures
Conclusion & Future Work

Conclusion

- Shown that https://github.com/fcpp portable & adaptable
- Discovered quite some variability in behaviour of Android phones
- Difficult to (globally) observe status of experiment through human proxies (even with central logging for debugging)

Future Work

- iOS-version, larger experiment, outdoors, ... to fine-tune comms-parameters & energy-consumption.
- Close the gaps between design, simulation and deployment.
- Formalization around spatio-temporal properties and their equivalences.
- Find partner in application domain.
Conclusion & Future Work

Conclusion

- Shown that https://github.com/fcpp portable & adaptable
- Discovered quite some variability in behaviour of Android phones
- Difficult to (globally) observe status of experiment through human proxies (even with central logging for debugging)

Future Work

- iOS-version, larger experiment, outdoors, ...
  - to fine-tune comms-parameters & energy-consumption.
- Close the gaps between design, simulation and deployment.
- Formalization around spatio-temporal properties and their equivalences.
- Find partner in application domain.

Thank You!