

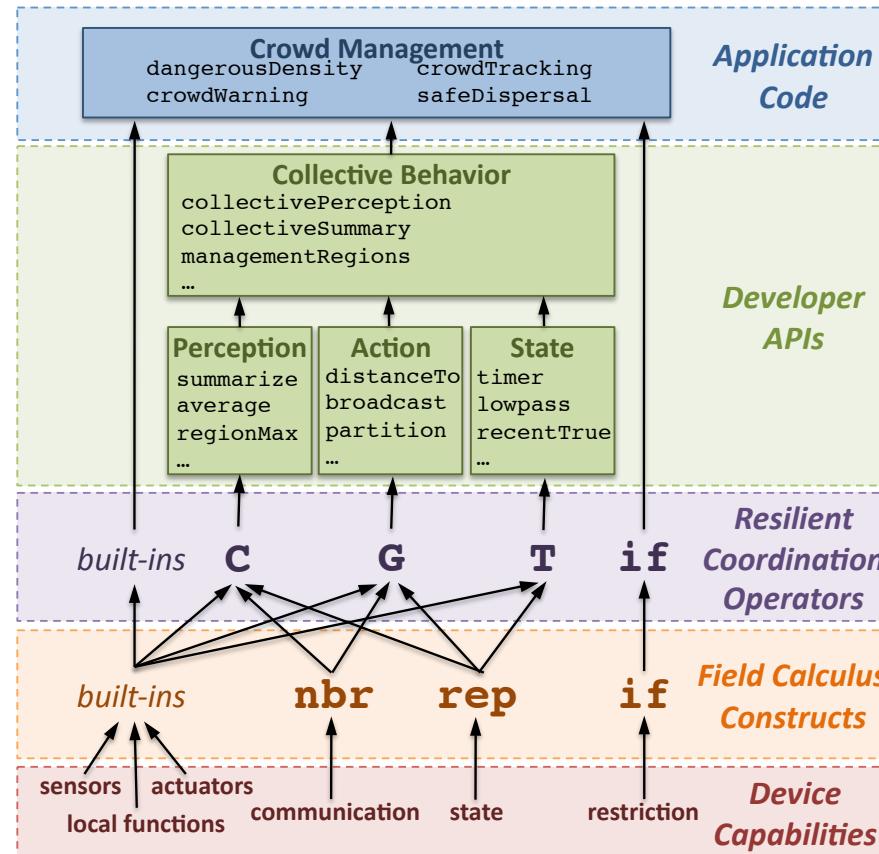
Ensuring Safe Composition of Distributed Processes

Jacob Beal

QA4SASO Workshop
@ IEEE SASO
September, 2015

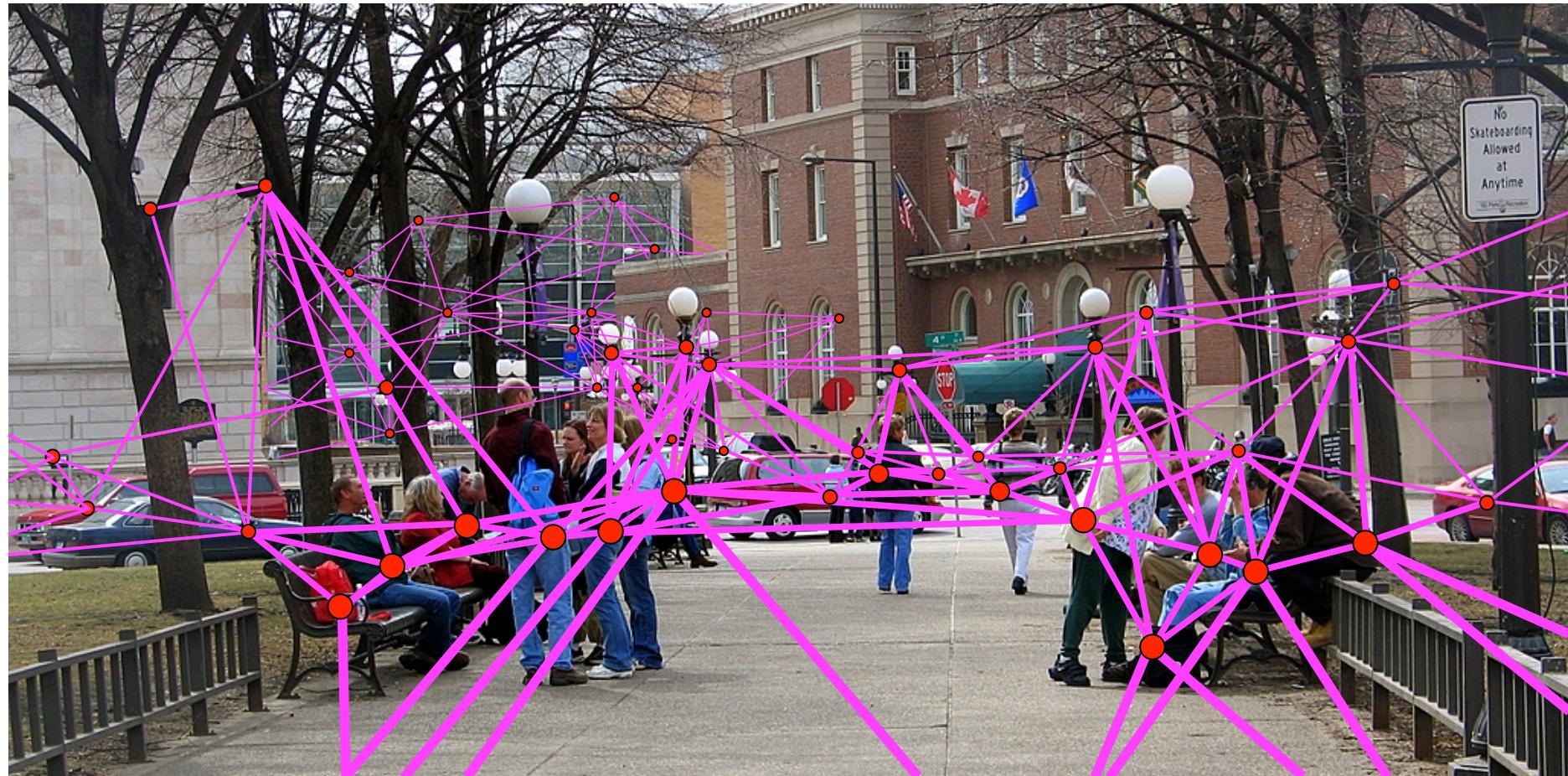
A generative approach to safety

Restrict your development environment...



... to contain only resilient distributed systems.

Everything is a wireless computer



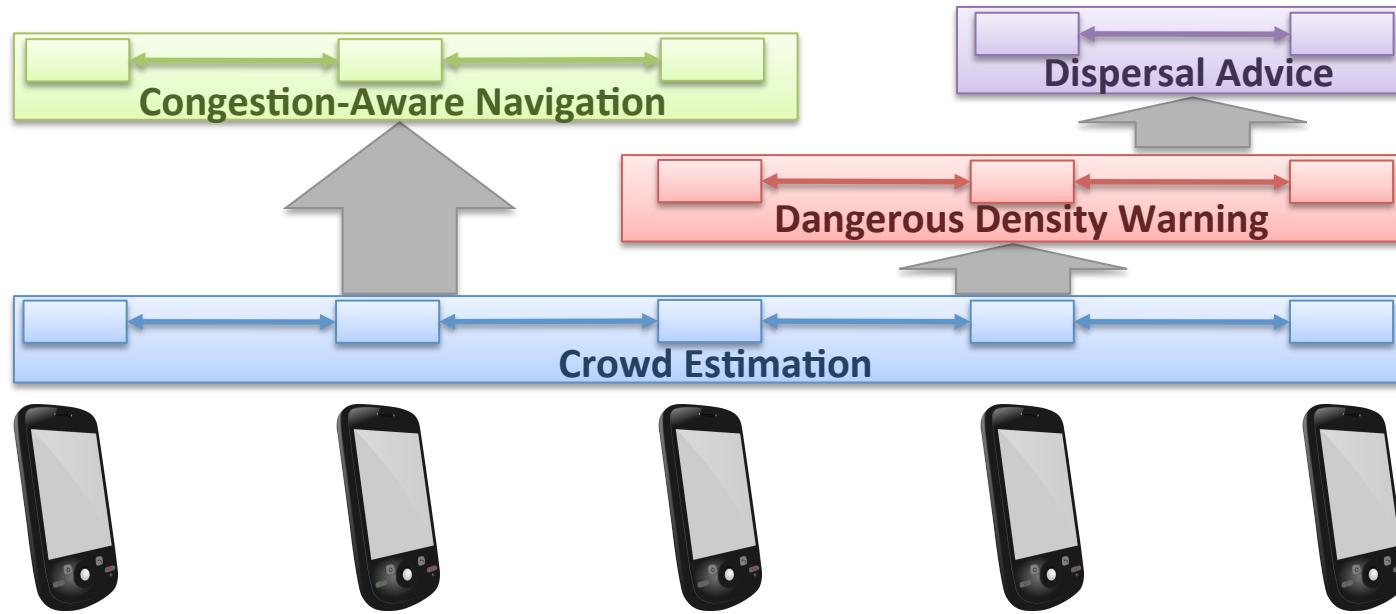
Example: Services for Mass Events



Example: Managing Crowd Danger



Distributed services are complex

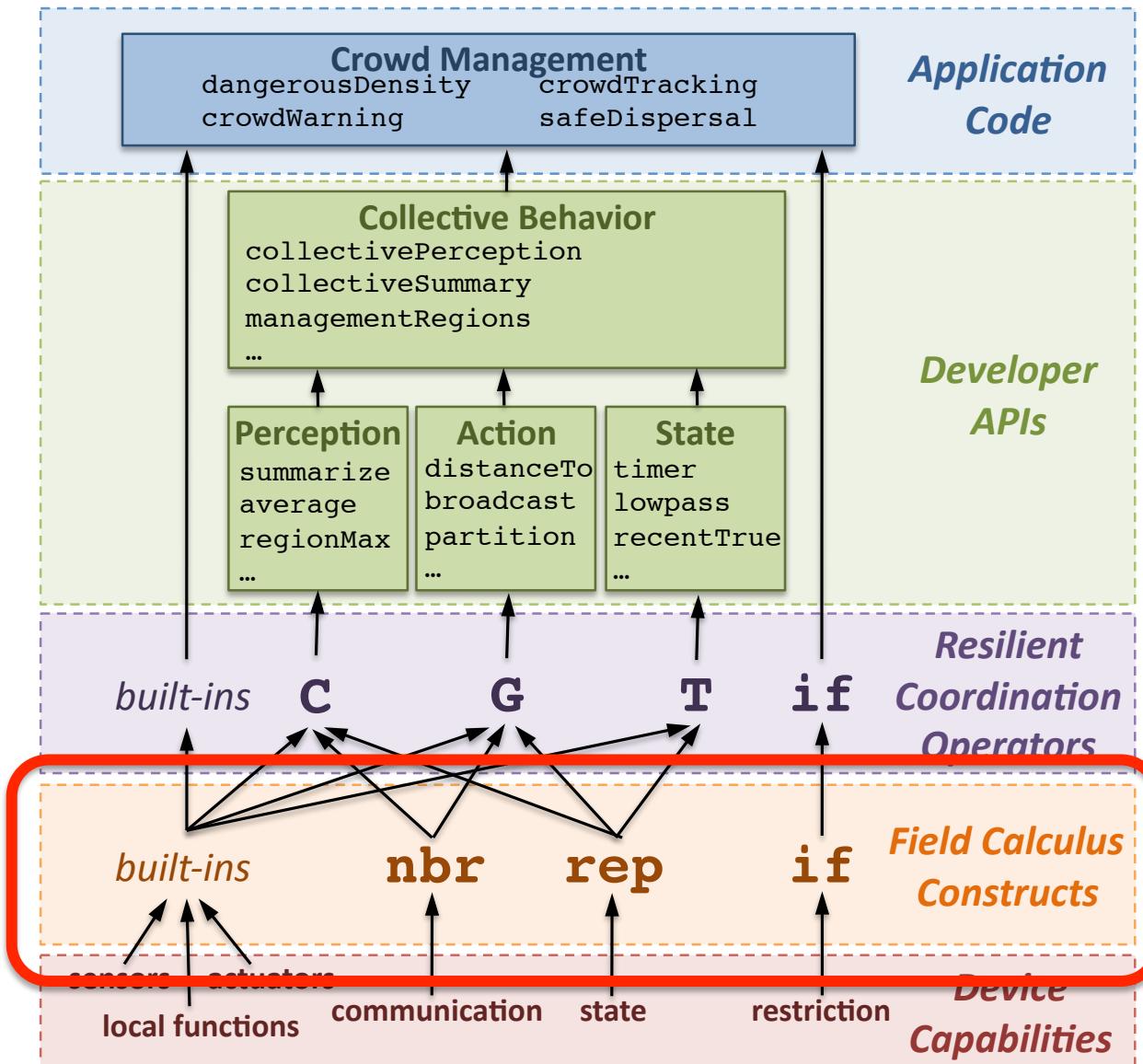


How to make engineering resilience tractable?

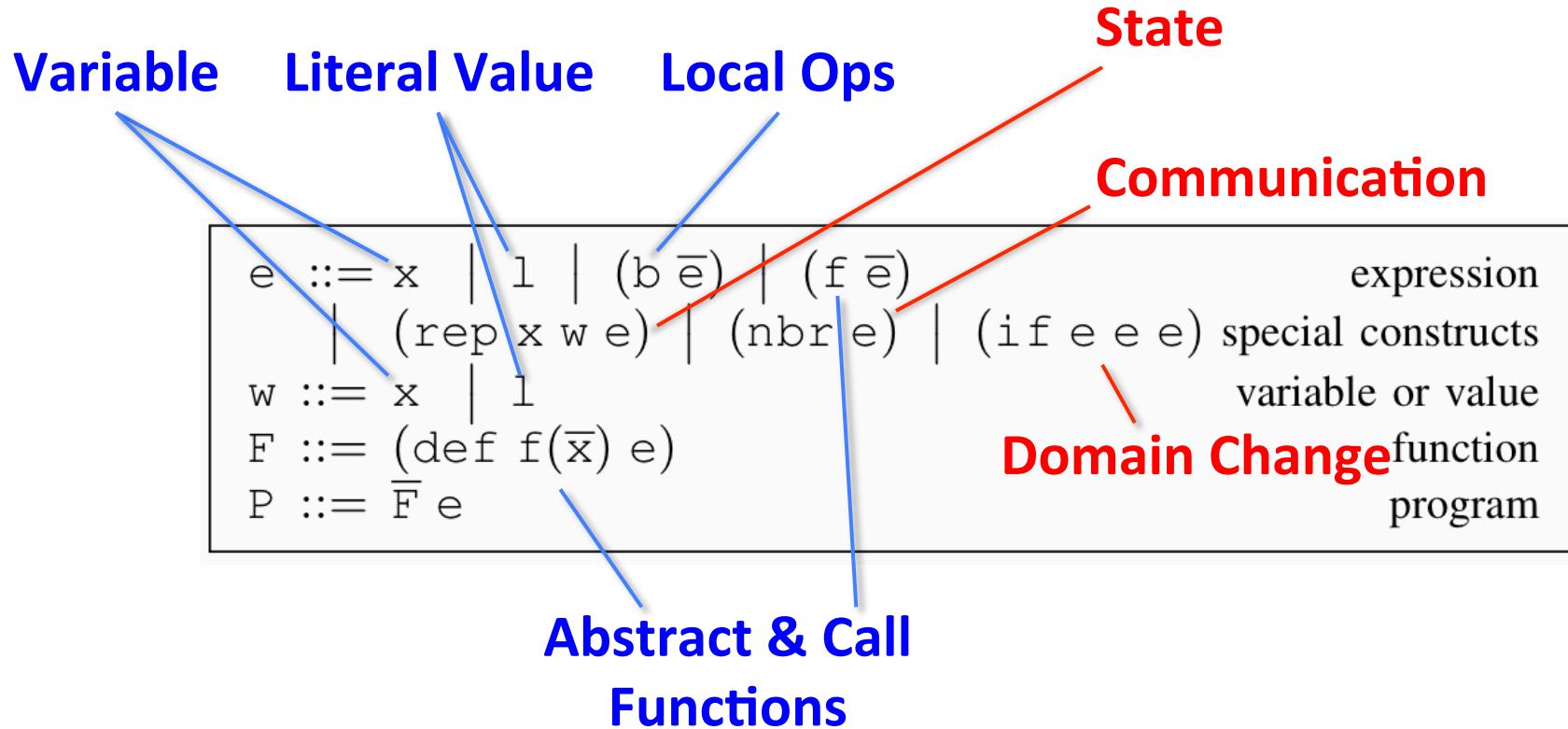
Aggregate Programming



Aggregate Programming Stack

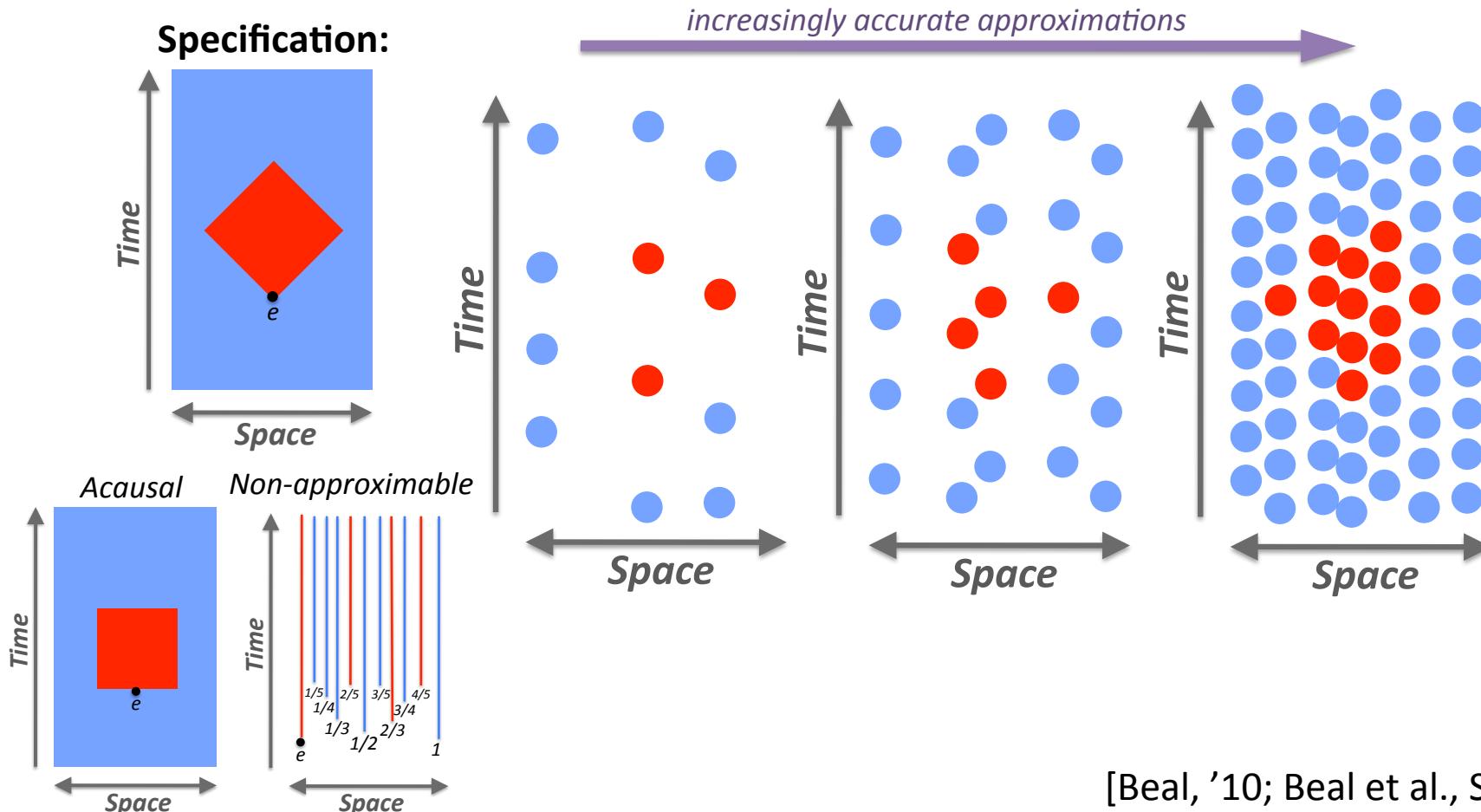


Field Calculus



Field Calculus is Space-Time Universal

Space-time Universal = arbitrarily good approximation of any causal, finitely-approximable computation

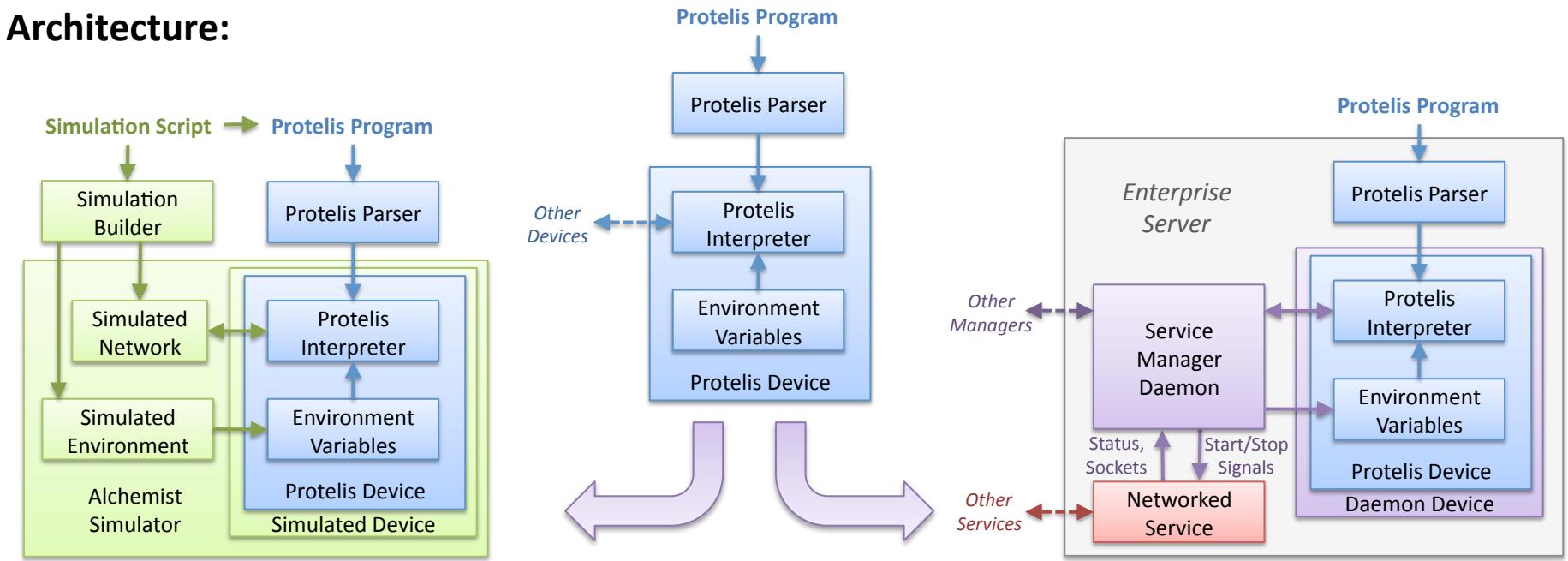


Instantiation: Protelis

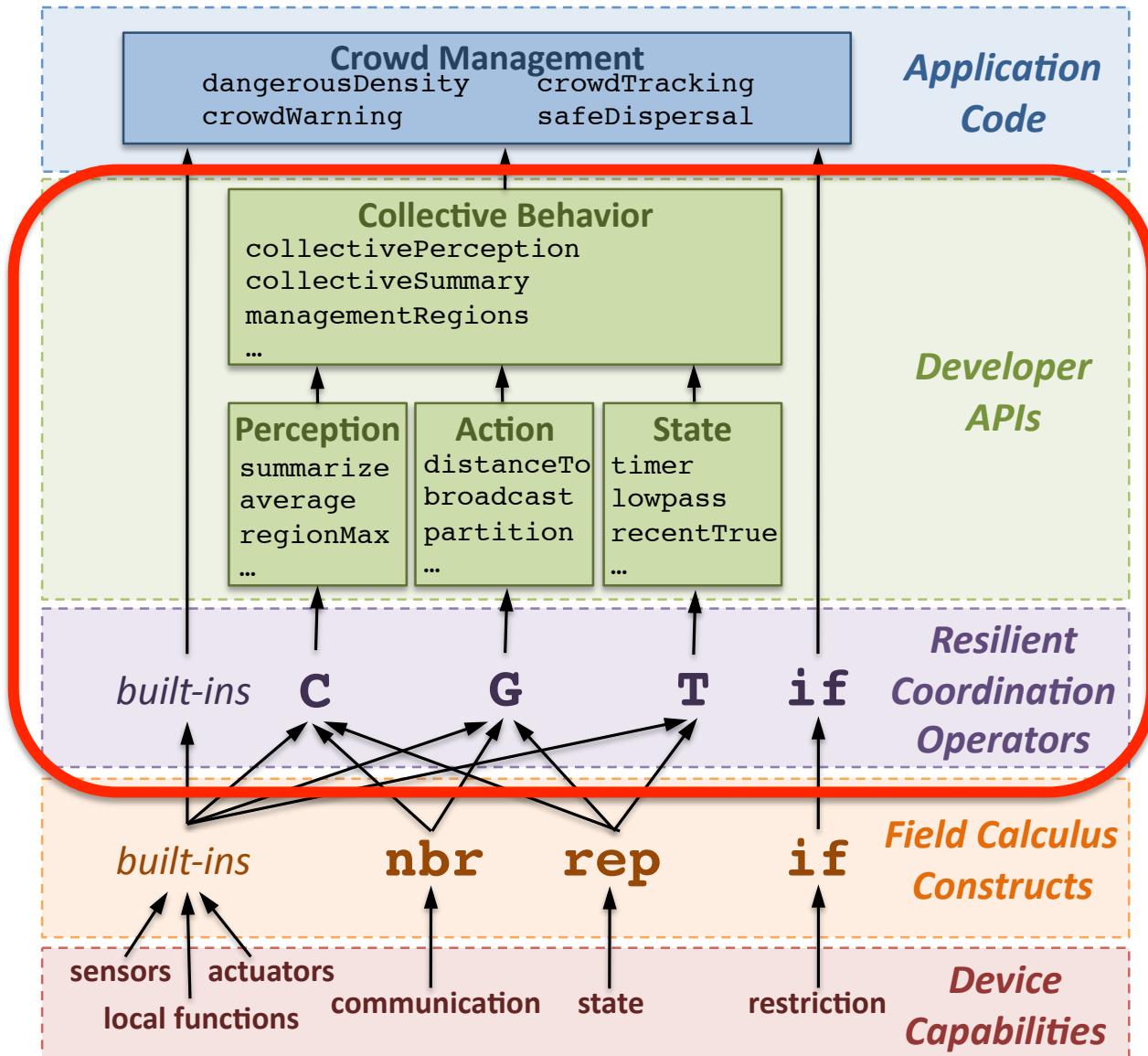
- Java-hosted & integrated
- Java-like syntax
- Eclipse support

```
def distanceTo(source) {
    rep(d <- Infinity) {
        mux (source) { 0 }
        else { minHood(nbr{d} + nbrRange) }
    }
}
```

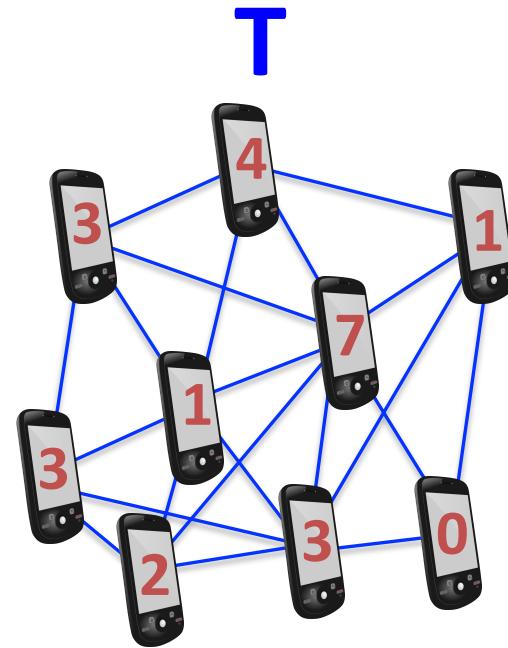
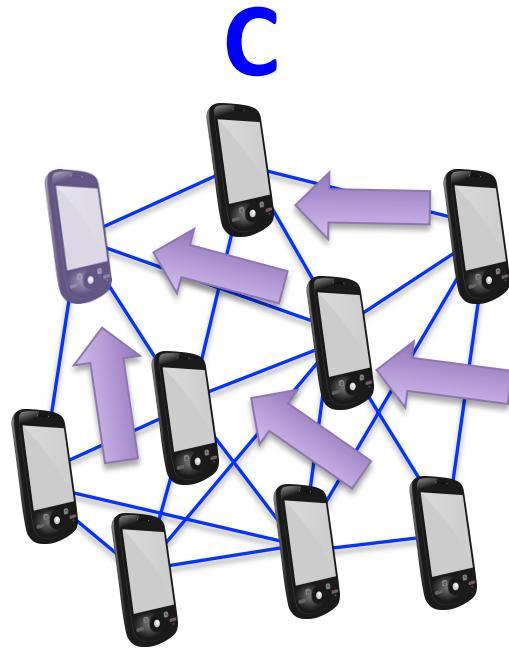
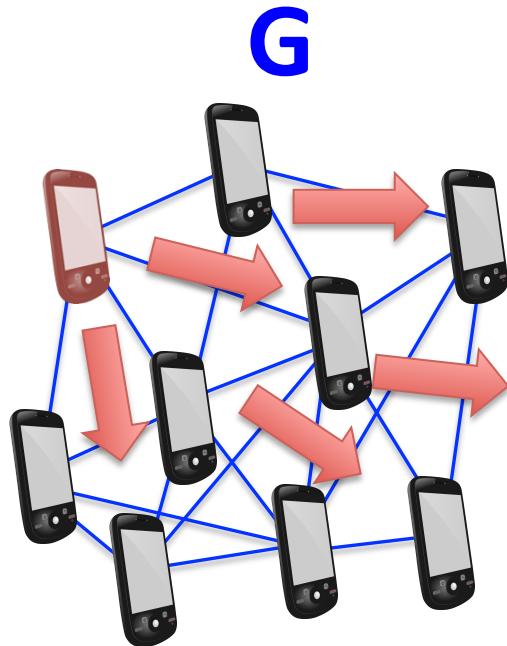
Architecture:



Aggregate Programming Stack



Self-Stabilizing Building Blocks



Information spreading Information collection Short-term memory

Resilience by construction: all programs from
these building blocks are also self-stabilizing!

Applying building blocks:

Example API algorithms from building blocks:

distance-to (source)	max-likelihood (source p)
broadcast (source value)	path-forecast (source obstacle)
summarize (sink accumulate local null)	average (sink value)
integral (sink value)	region-max (sink value)
timer (length)	limited-memory (value timeout)
random-voronoi (grain metric)	group-size (region)
broadcast-region (region source value)	recent-event (event timeout)
distance-avoiding-obstacles (source obstacles)	

Since based on these building blocks, all programs built this way are self-stabilizing!

Complex Example: Crowd Management

```
(def crowd-tracking (p)
  ;; Consider only Fruin LoS E or F within last minute
  (if (recently-true (> (density-est p) 1.08) 60)
    ;; Break into randomized "cells" and estimate danger of each
    (+ 1 (dangerous-density (sparse-partition 30) p))
    0))

(def recently-true (state memory-time)
  ;; Make sure first state is false, not true...
  (rt-sub (not (T 1 1)) state memory-time))
(def rt-sub (started s m)
  (if state 1 (limited-memory s m)))

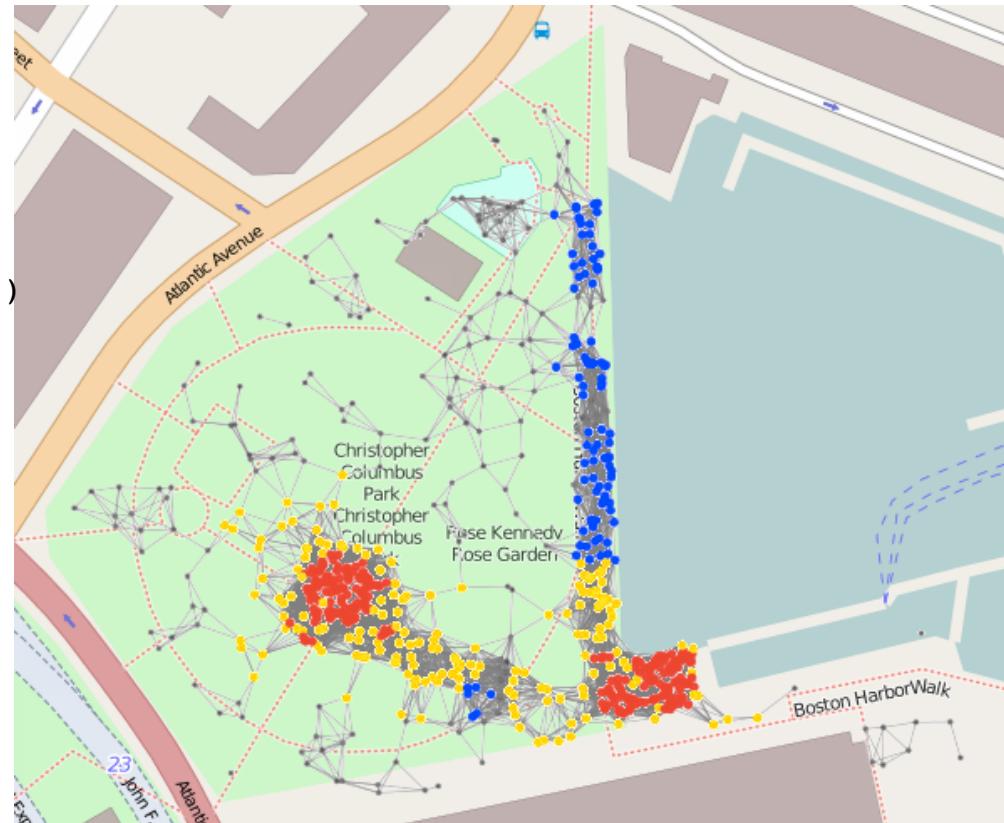
(def dangerous-density (partition p)
  ;; Only dangerous if above critical density threshold...
  (and
    (> (average partition (density-est p)) 2.17)
    ;; ... and also involving many people.
    (> (summarize partition + (/ 1 p) 0) 300)))
```

*18 lines non-whitespace code
10 library calls (21 ops)*

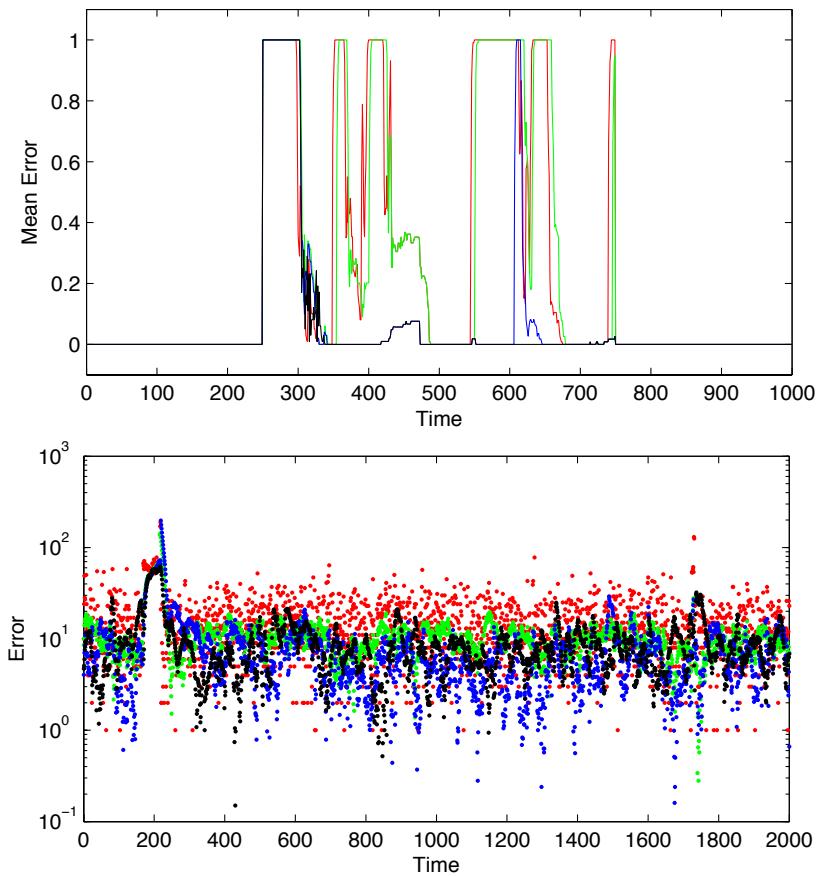
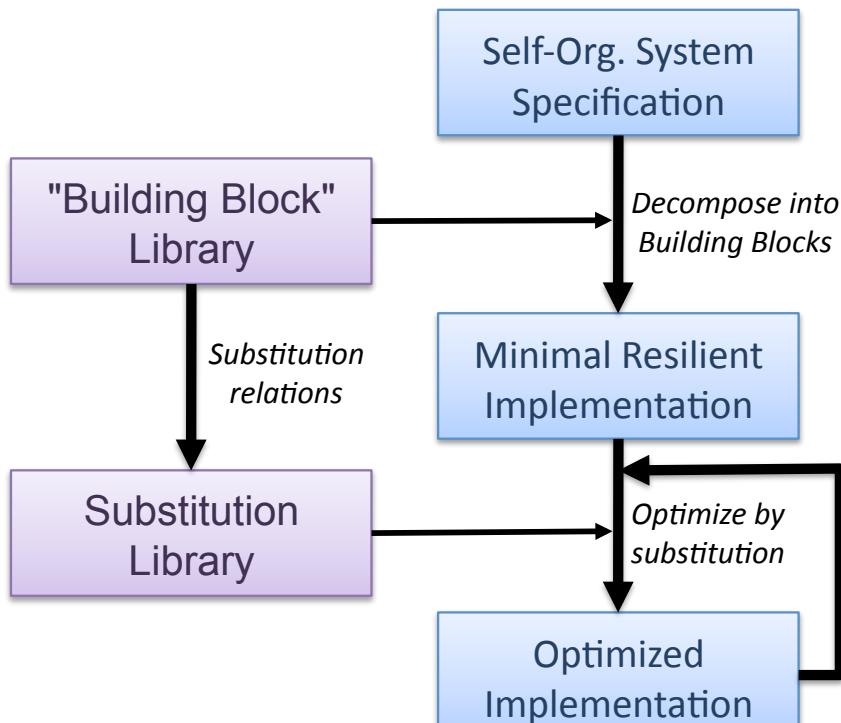
IF: 3 G: 11 C: 4 T: 3

```
(def crowd-warning (p range)
  (> (distance-to (= (crowd-tracking p) 2))
      range))

(def safe-navigation (destination p)
  (distance-avoiding-obstacles
    destination (crowd-warning p)))
```



Optimization of Dynamics

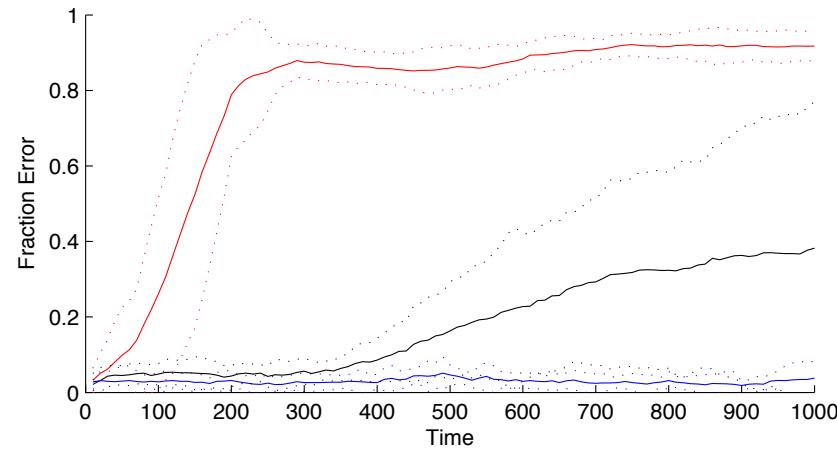
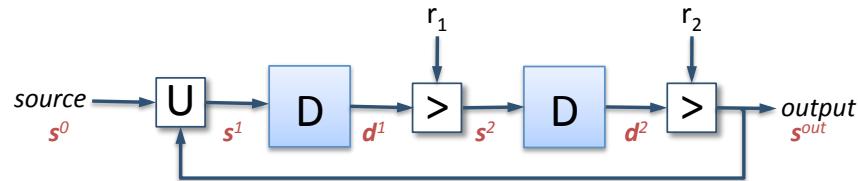
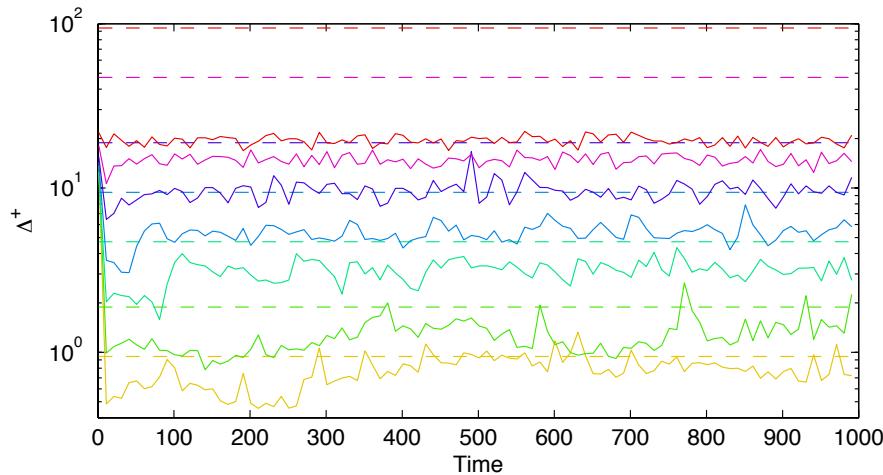


See our talk in the main program of SASO!

Predicting and Controlling Dynamics

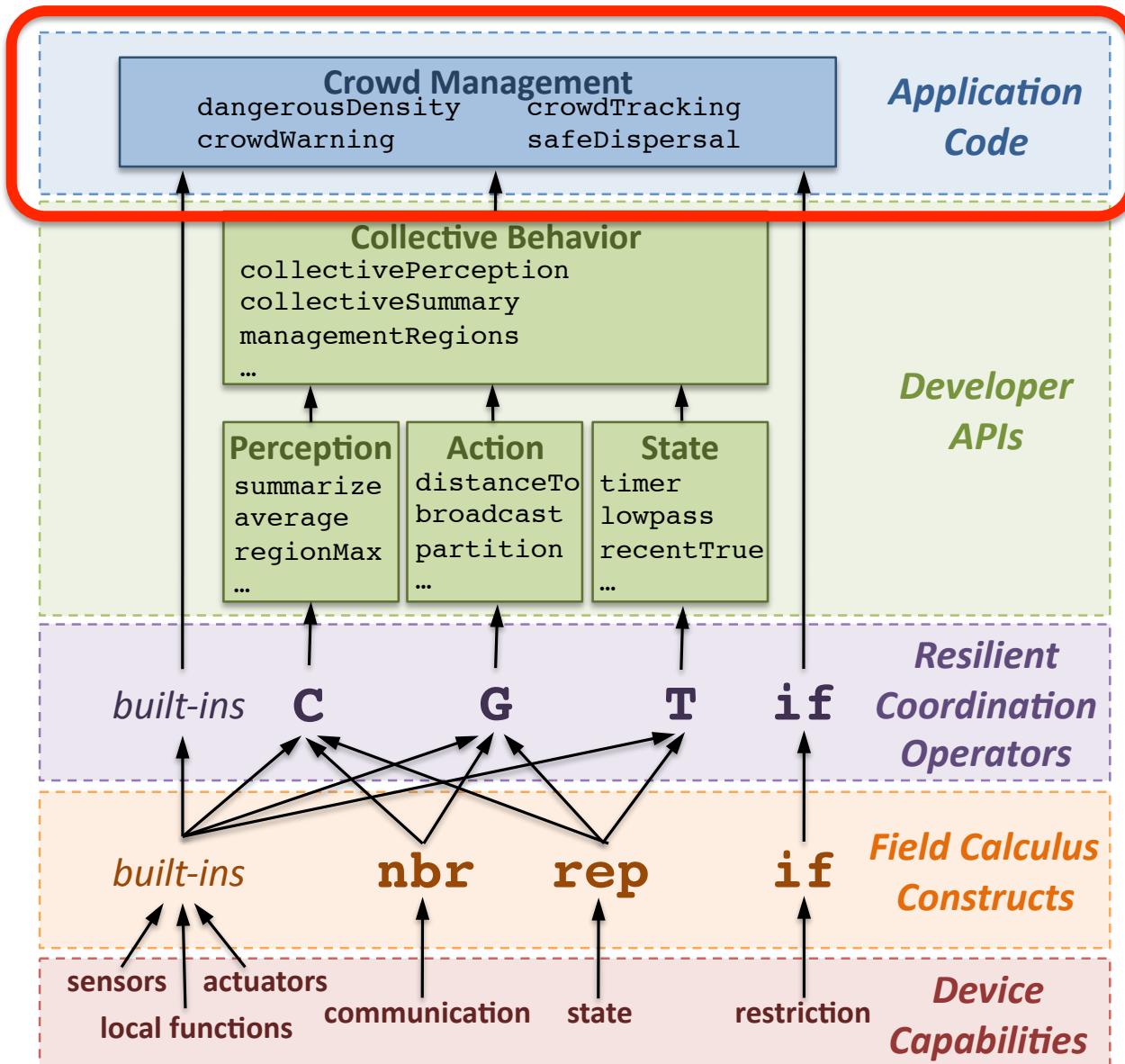
Distance estimate: prediction from transient response:

$$\Delta^+[t] \rightarrow \frac{d\Delta^+[t]}{dt} \cdot \text{diameter}$$

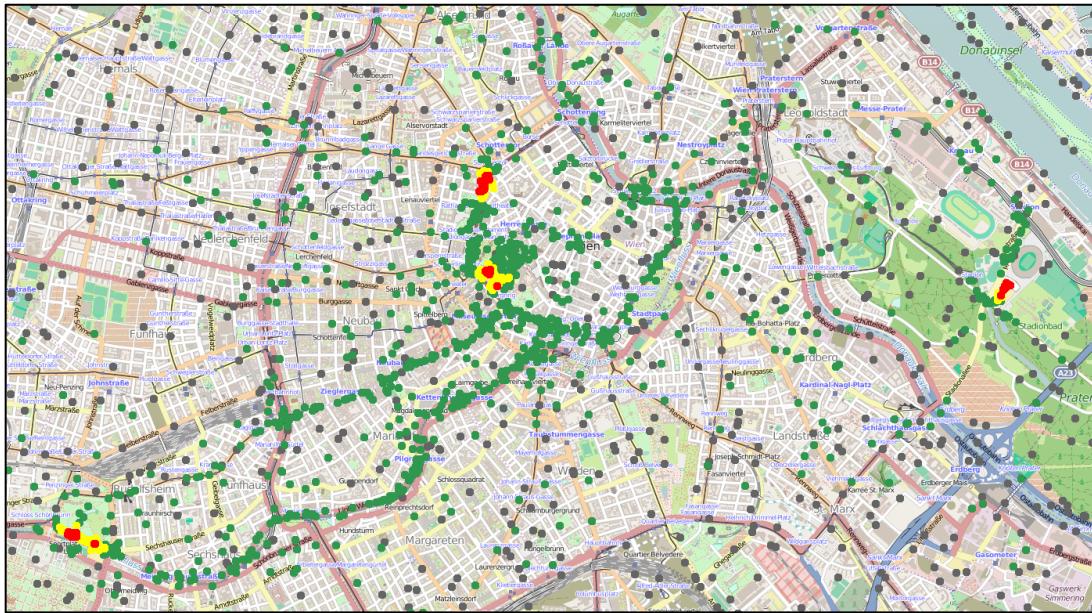


See our talk in the FOCAS/SCOPES workshop!

Aggregate Programming Stack



Crowd Safety Services

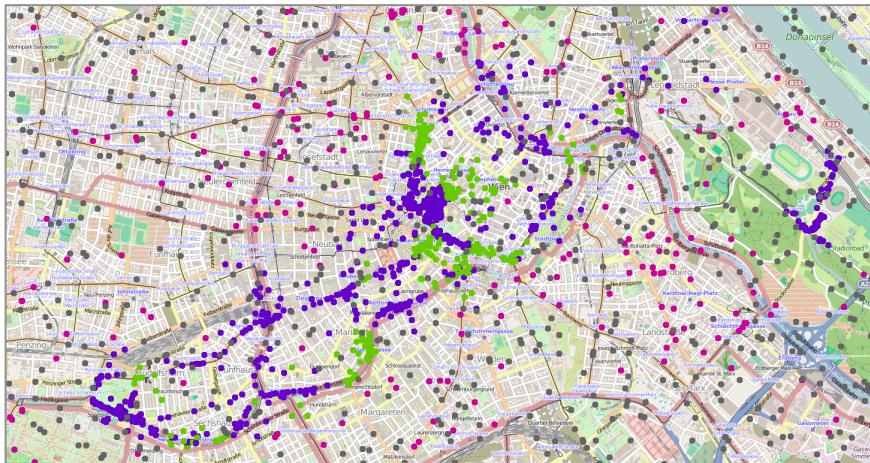


```

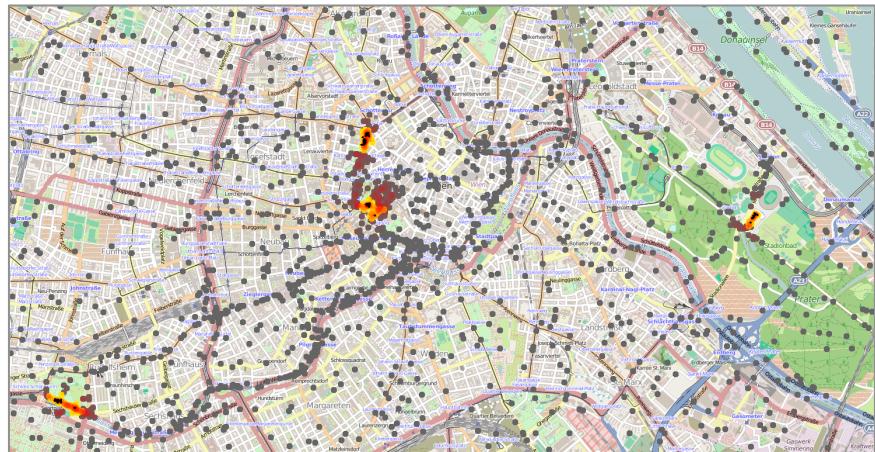
def dangerousDensity(p, r) {
    let mr = managementRegions(r*2, () -> { nbrRange });
    let danger = average(mr, densityEst(p, r)) > 2.17 &&
        summarize(mr, sum, 1 / p, 0) > 300;
    if(danger) { high } else { low }
}
def crowdTracking(p, r, t) {
    let crowdRgn = recentTrue(densityEst(p, r)>1.08, t);
    if(crowdRgn) { dangerousDensity(p, r) } else { none };
}
def crowdWarning(p, r, warn, t) {
    distanceTo(crowdTracking(p,r,t) == high) < warn
}

```

Dissemination of new versions

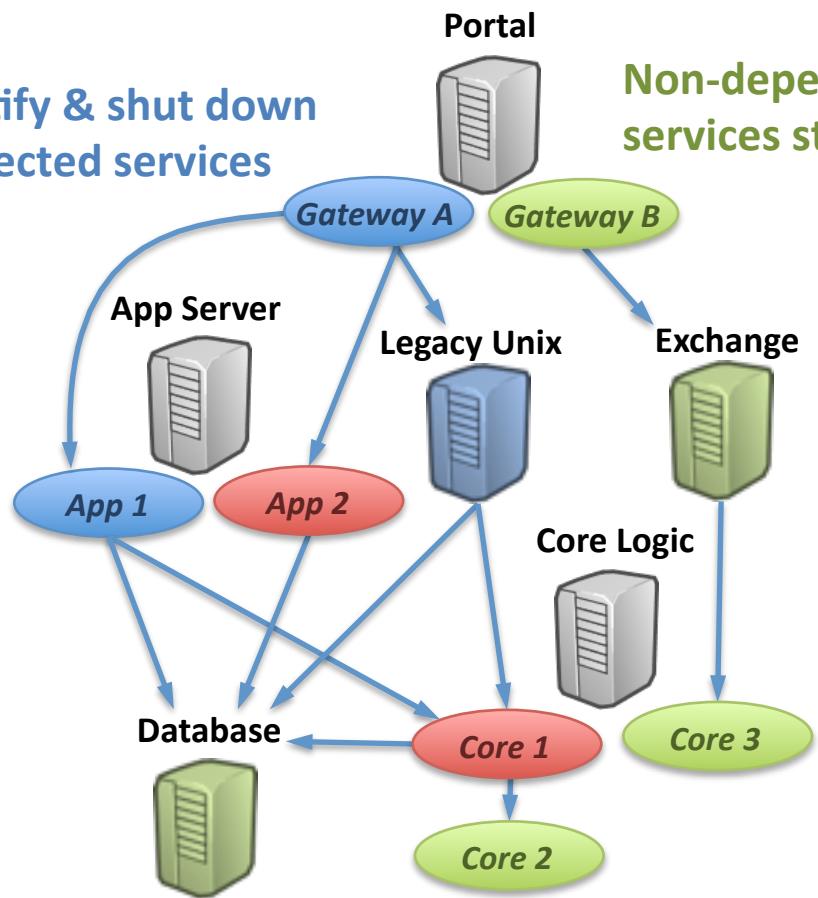


Pre-emptive modulation of priorities



Dependency-Directed Recovery

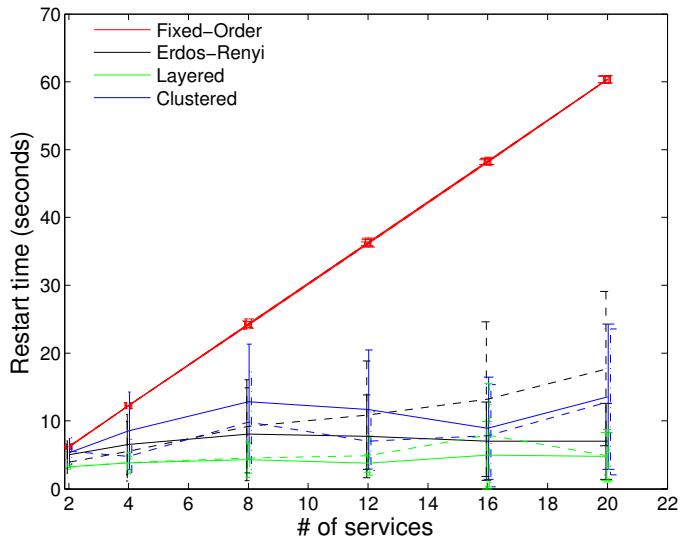
Identify & shut down
affected services



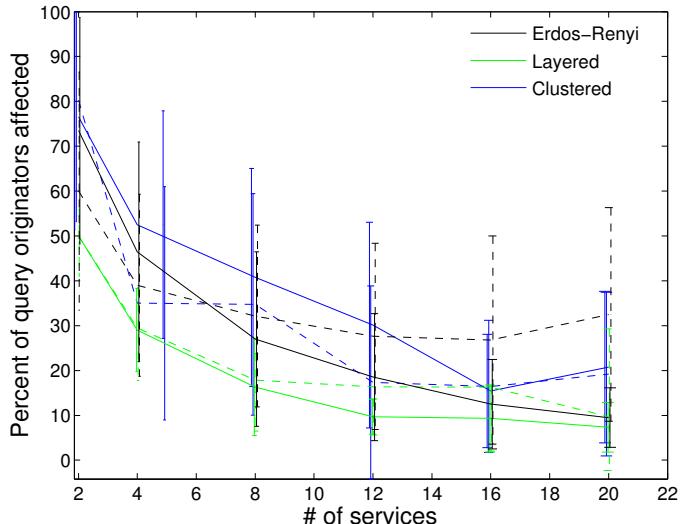
Non-dependent
services still run

See our talk in the main
program of SASO!

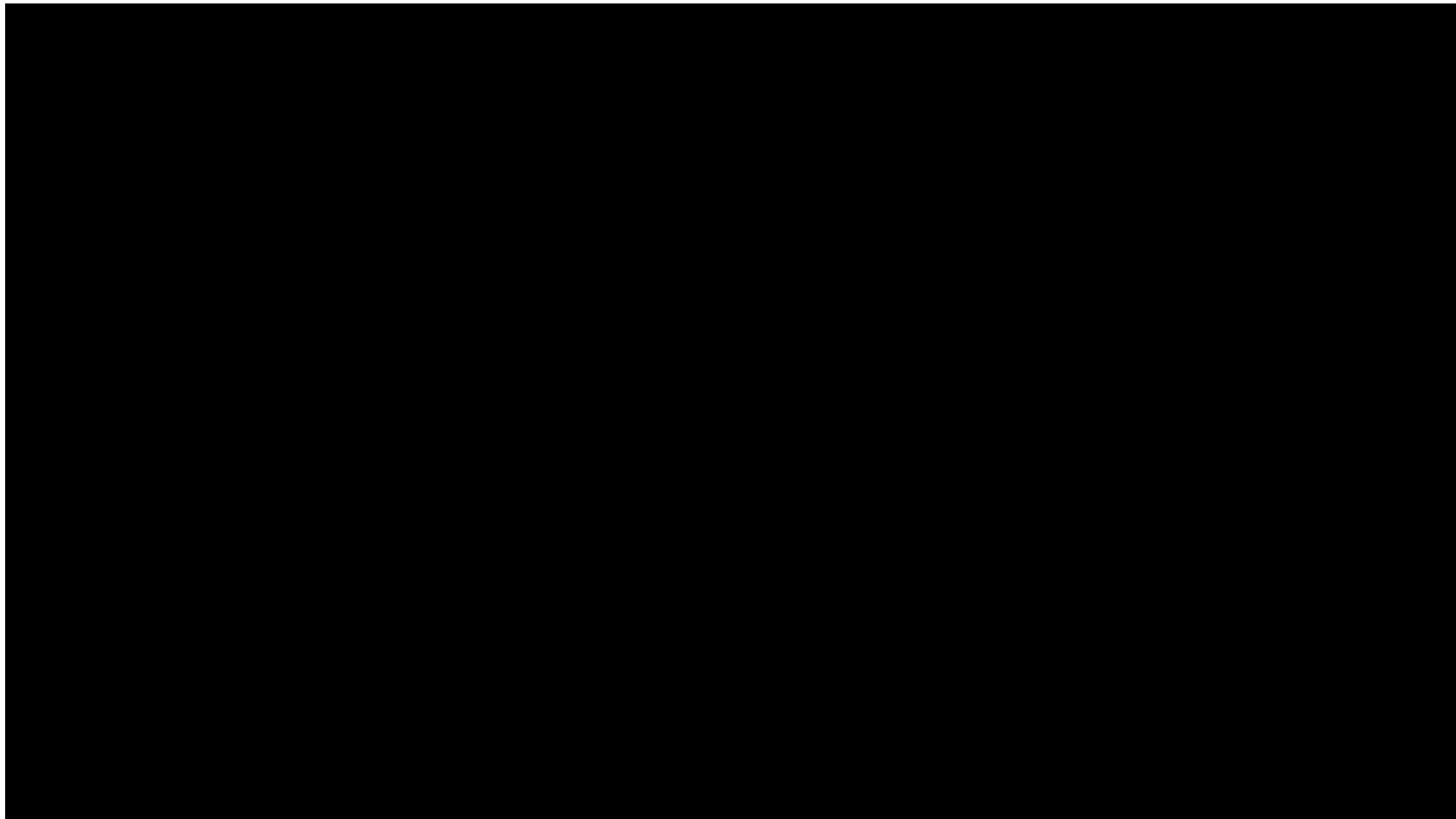
Dramatically better recovery time



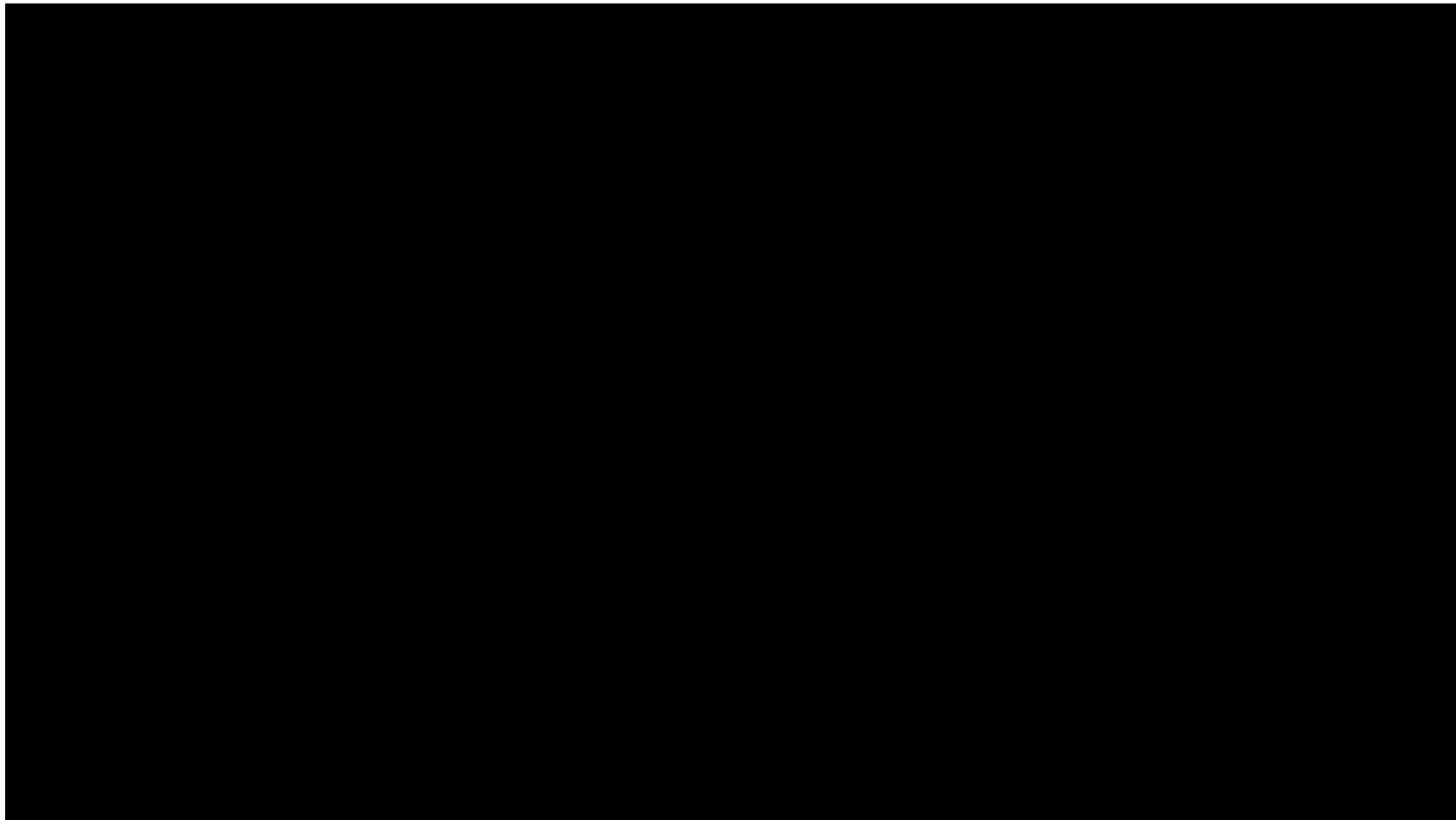
Fewer services disrupted



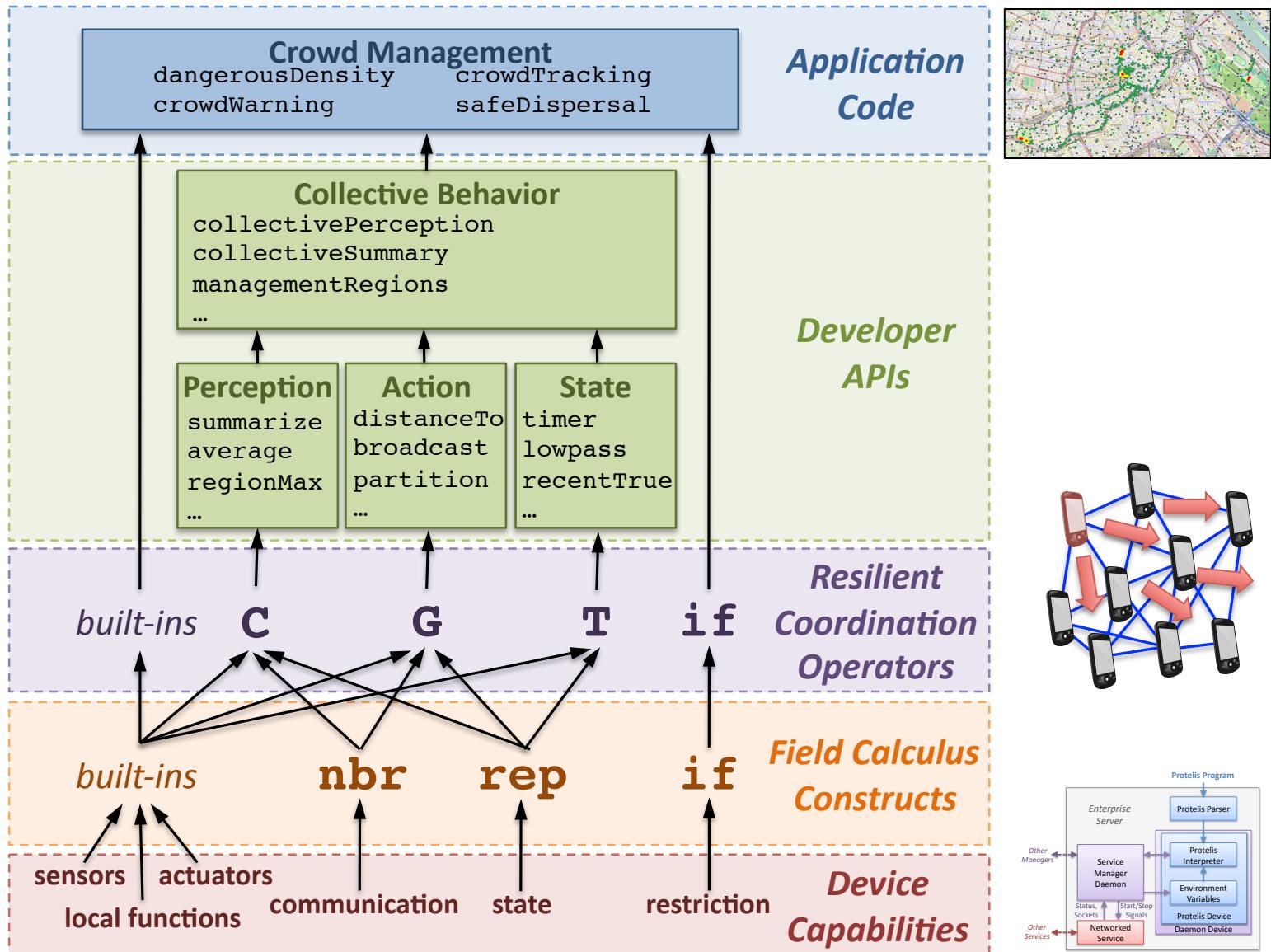
Distributed Rewind and Replay



Tactical Cloud Services



Summary: Aggregate Methodology



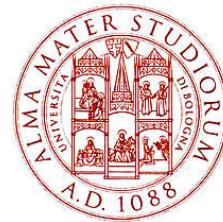
Acknowledgements

Raytheon **BBN Technologies**

- Shane Clark
- Partha Pal
- Kyle Usbeck



- Soura Dasgupta
- Raghu Mudumbai
- Amy Kumar



- Mirko Viroli
- Danilo Pianini



- Ferruccio Damiani

