Aggregate Programming
Part 3: Application Examples

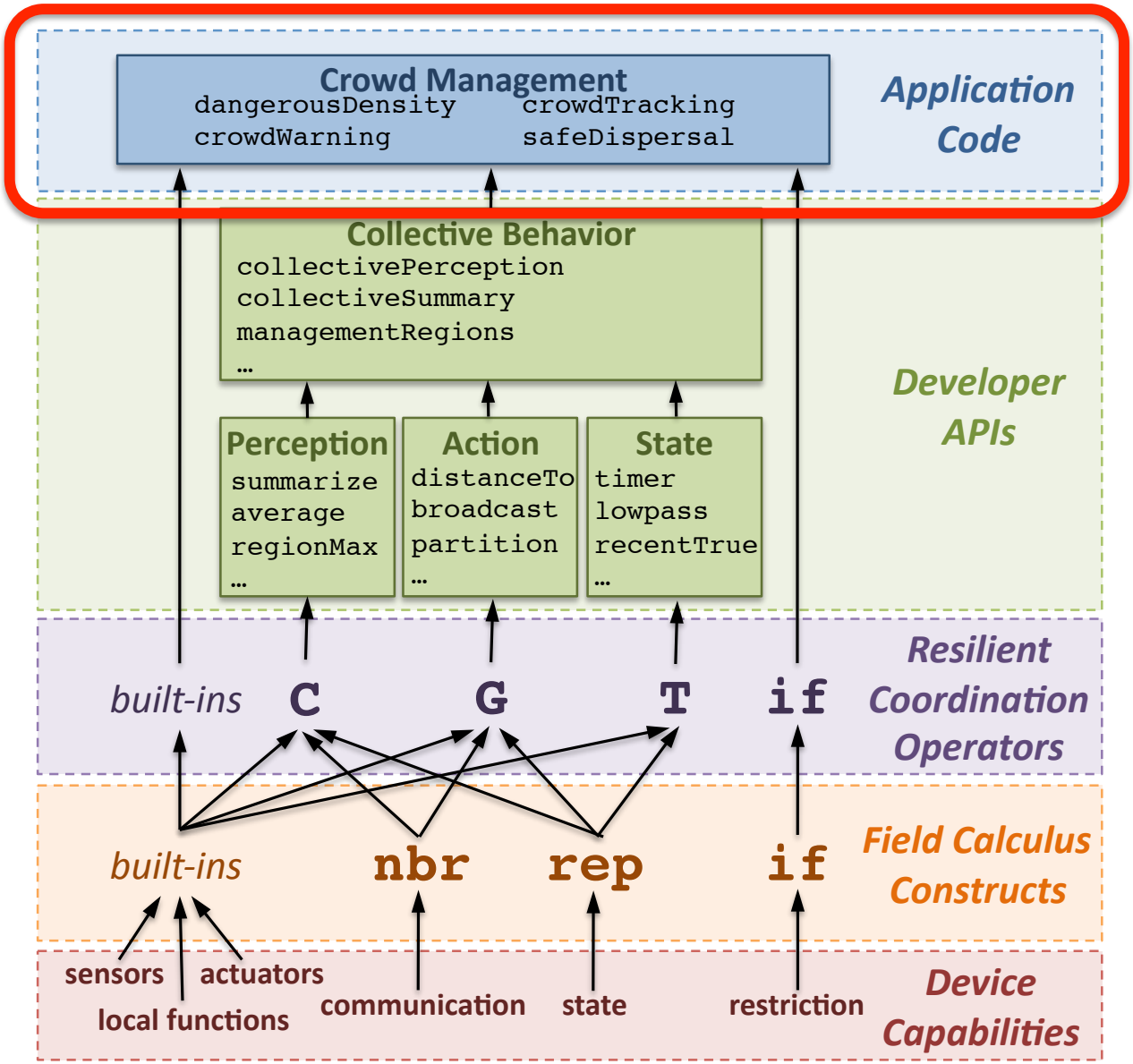
Jacob Beal

SFM16: QUANTICOL
June 2016

Raytheon
BBN Technologies

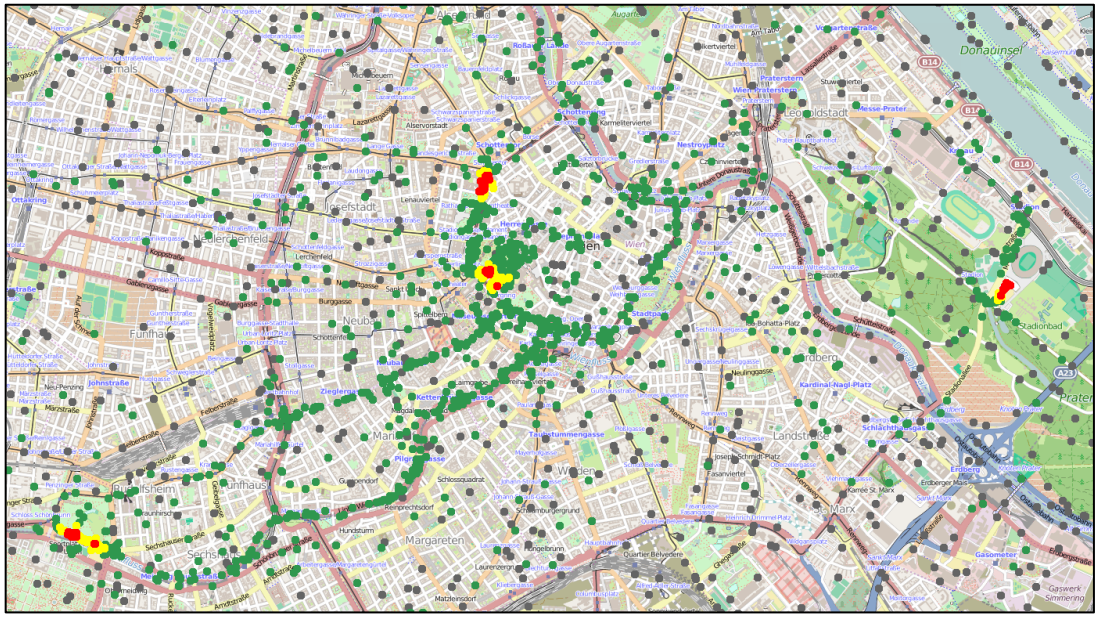
Aggregate Programming Stack

Raytheon BBN Technologies



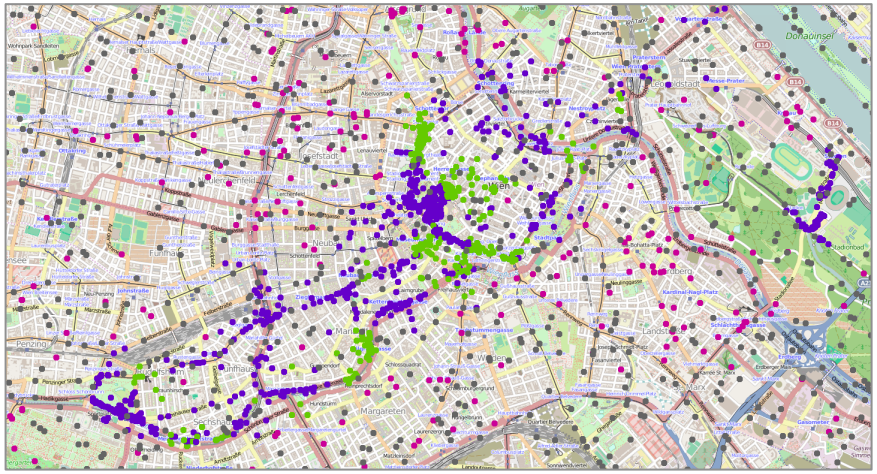
Crowd Safety Services

Raytheon
BBN Technologies

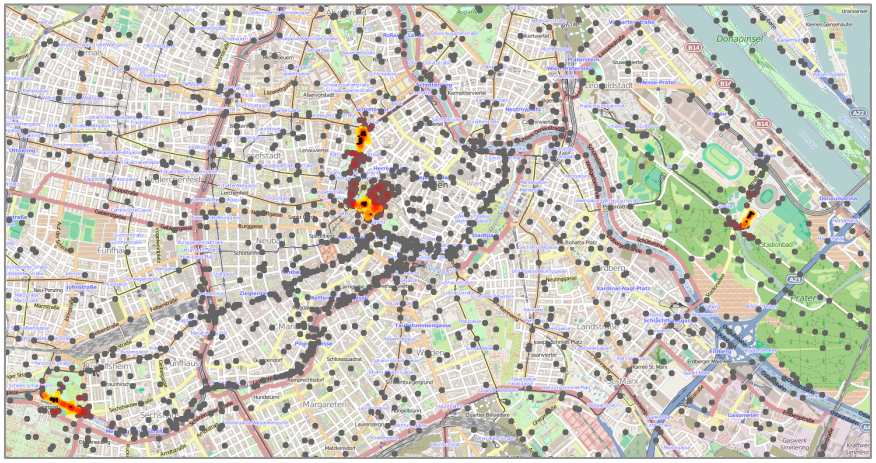


```
def dangerousDensity(p, r) {
  let mr = managementRegions(r*2, () -> { nbrRange });
  let danger = average(mr, densityEst(p, r)) > 2.17 &&
               summarize(mr, sum, 1 / p, 0) > 300;
  if(danger) { high } else { low }
}
def crowdTracking(p, r, t) {
  let crowdRgn = recentTrue(densityEst(p, 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

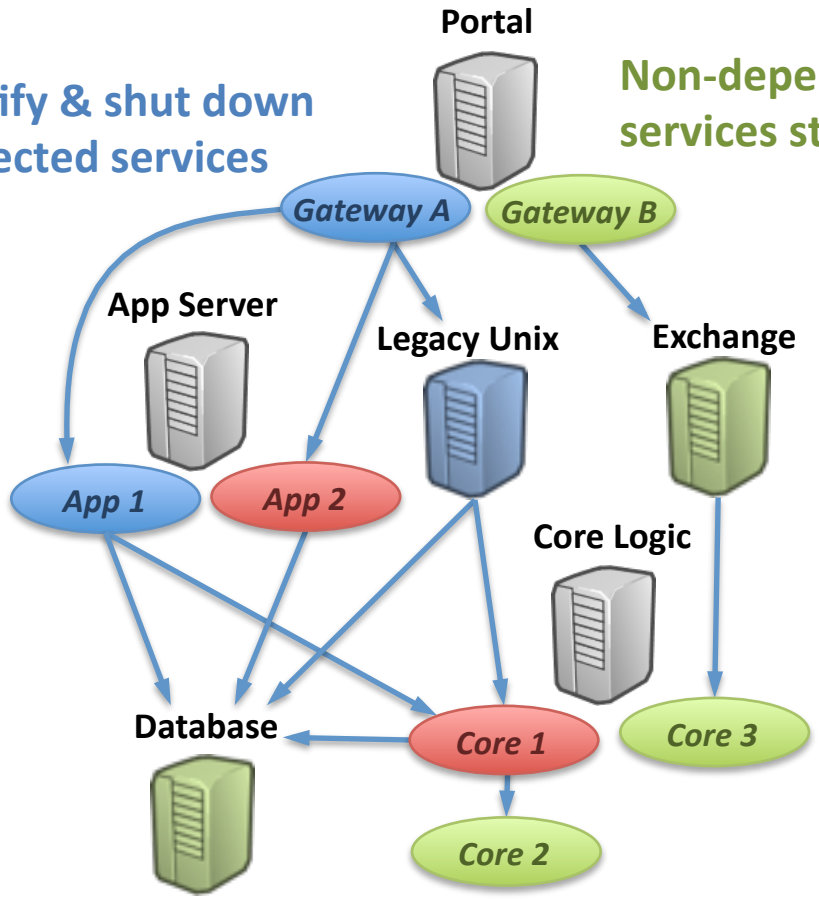


Pre-emptive modulation of priorities

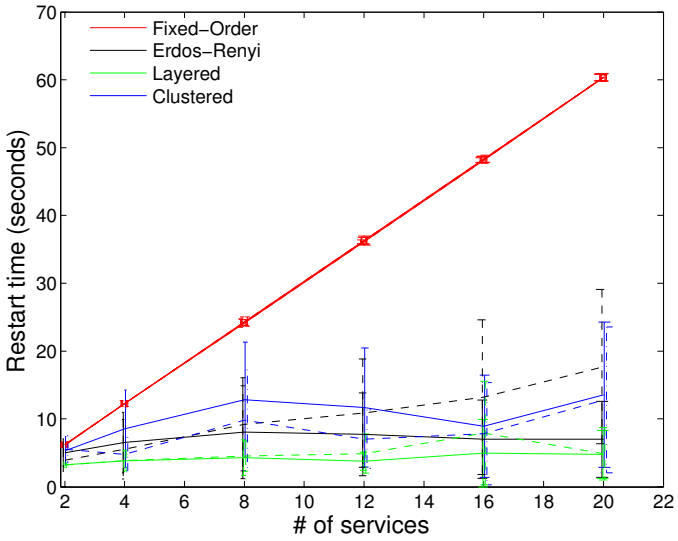


Dependency-Directed Recovery

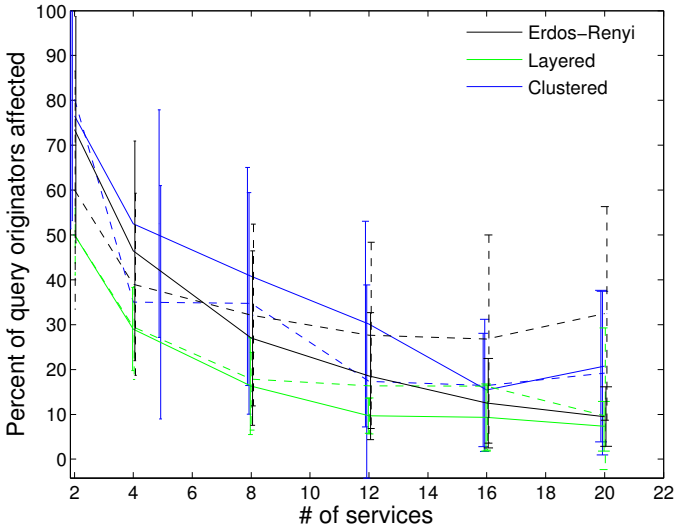
Raytheon BBN Technologies



Dramatically better recovery time



Fewer services disrupted



[Clark, Beal, & Pal, SASO '15]

DDR Algorithm

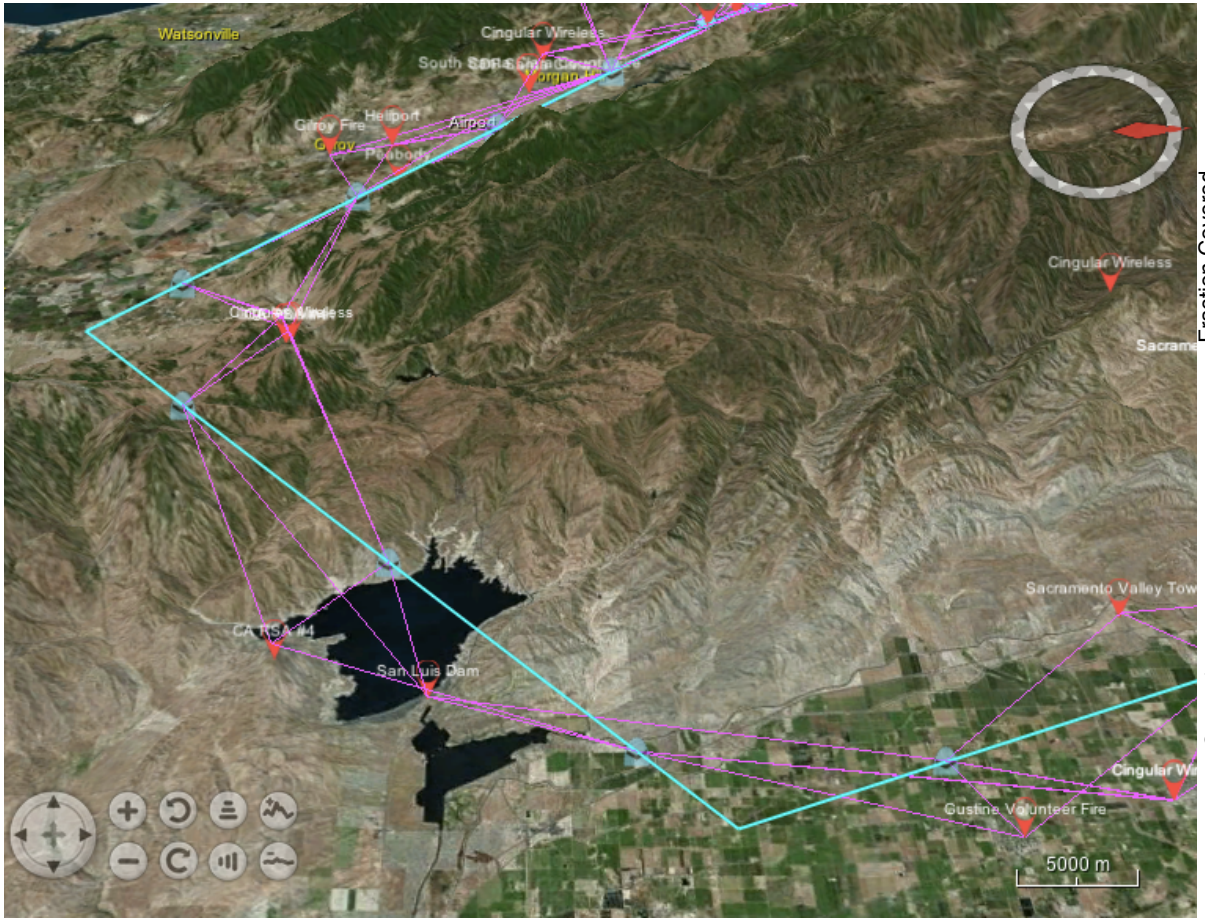
Raytheon
BBN Technologies

```
// Collect state of monitored service from service manager daemon
let status = self.getEnvironmentVariable("serviceStatus");
let serviceID = self.getEnvironmentVariable("serviceID");
let depends = self.getEnvironmentVariable("dependencies");
let serviceDown = status=="hung" || status=="stop";

// Compute whether service can safely be run (i.e. dependencies are satisfied)
let liveSet = if(serviceDown) { [] } else { [serviceID] };
let nbrsLive = unionHood(nbr(liveSet));
let liveDependencies = nbrsLive.intersection(depends);
let safeToRun = liveDependencies.equals(depends);

// Act based on service state and whether it is safe to run
if(!safeToRun) {
  if(!serviceDown) {
    self.stopService() // Take service down to avoid misbehavior
  } else { false } // Wait for dependencies to recover before restarting
} else {
  if(serviceDown) {
    self.restartService() // Safe to restart
  } else { false } // Everything fine; no action needed
}
```

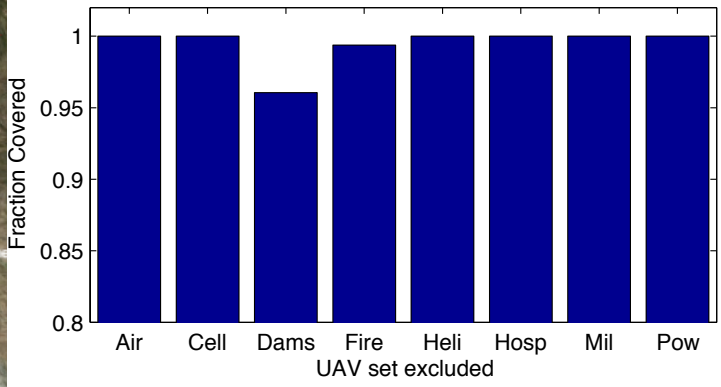
Opportunistic Airborne Sensor Sharing

Raytheon BBN Technologies

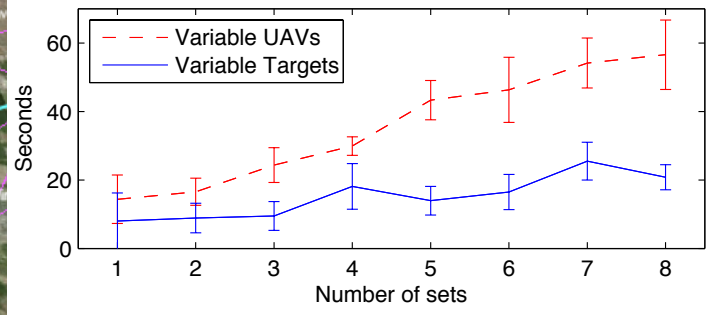
GIS-integrated adaptive mission planning



Highly effective sharing



Low computational cost



[Beal, Usbeck, Loyall & Metzler, *in prep*]

Distributed Rewind and Replay

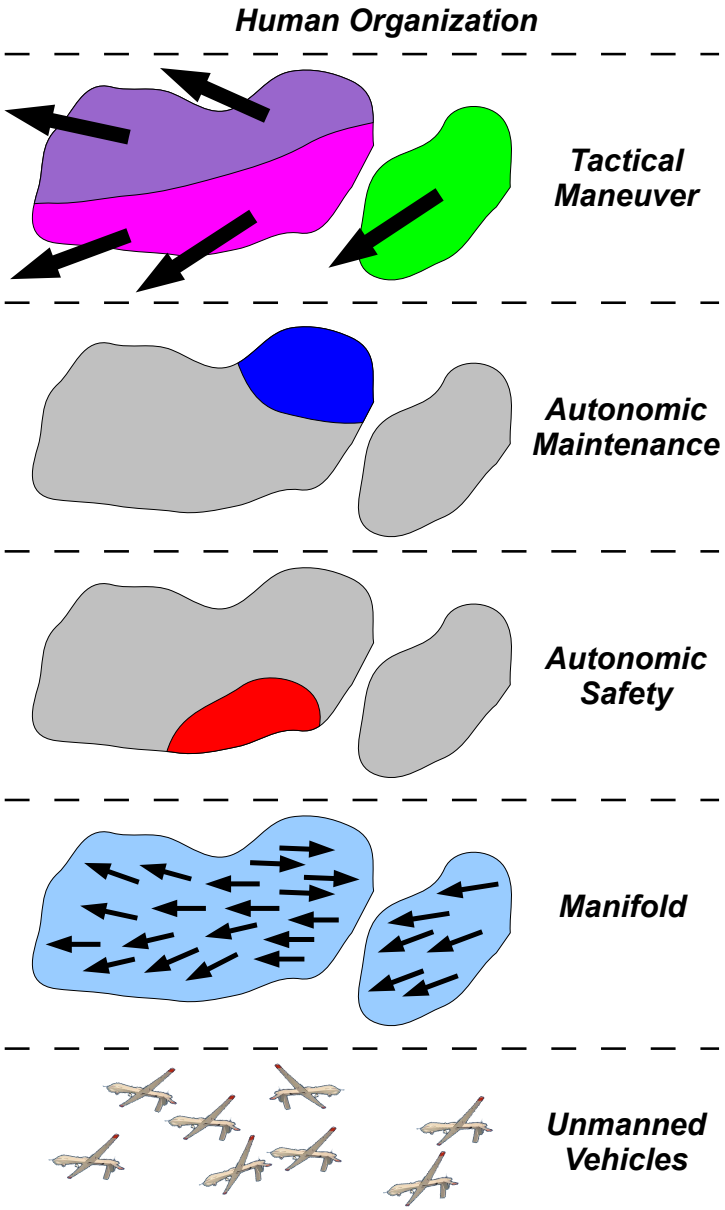
Raytheon
BBN Technologies

Tactical Cloud Services

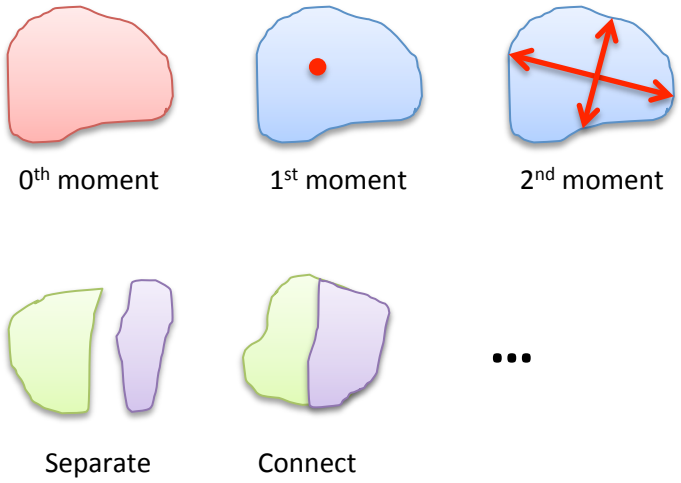
Raytheon
BBN Technologies

Swarm Command & Control

Raytheon BBN Technologies



- Build on top of continuous space model with:
  - Modulation by autonomic management layers
  - Basis set of "maneuvers"



[Beal, 2012]

Agent-Based Simulations

Raytheon
BBN Technologies

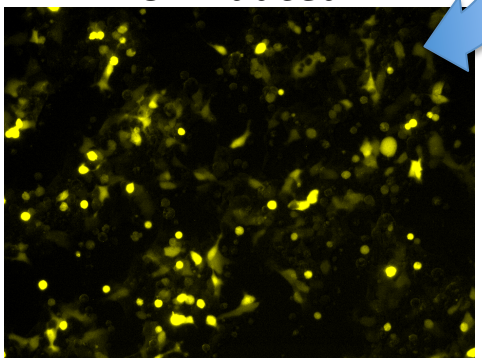
[Beal et al., 2012]

Engineering Biological Systems

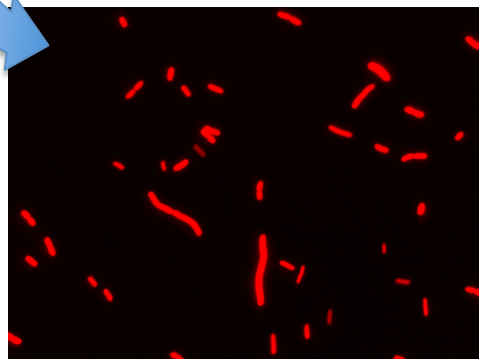
Raytheon BBN Technologies

```
(def simple-sensor-actuator ()
  (let ((x (test-sensor)))
    (debug-1 x)
    (debug-2 (not x))))
```

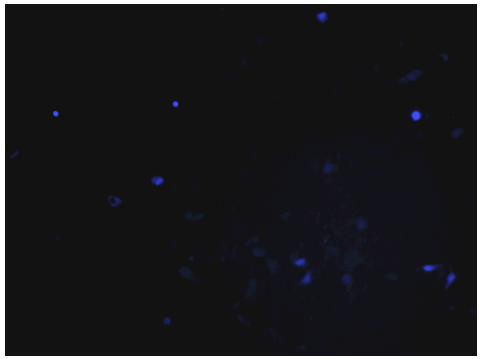
Uninduced



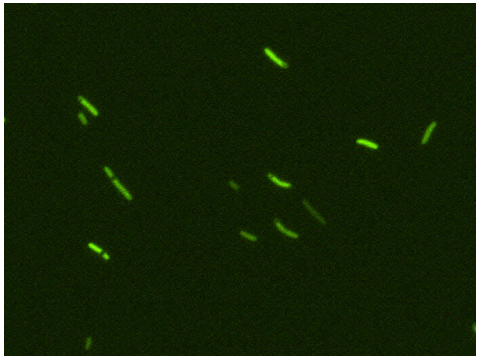
Uninduced



Induced

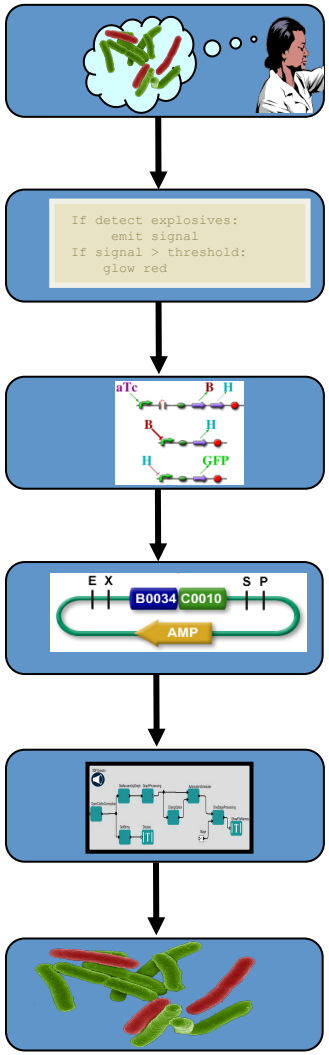


Induced



Mammalian Target

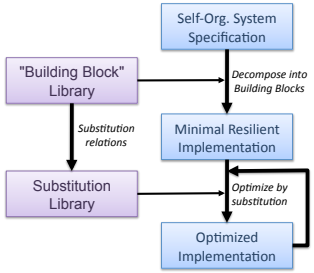
E. coli Target

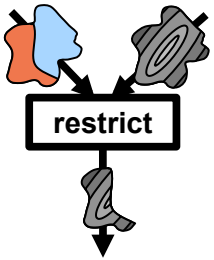


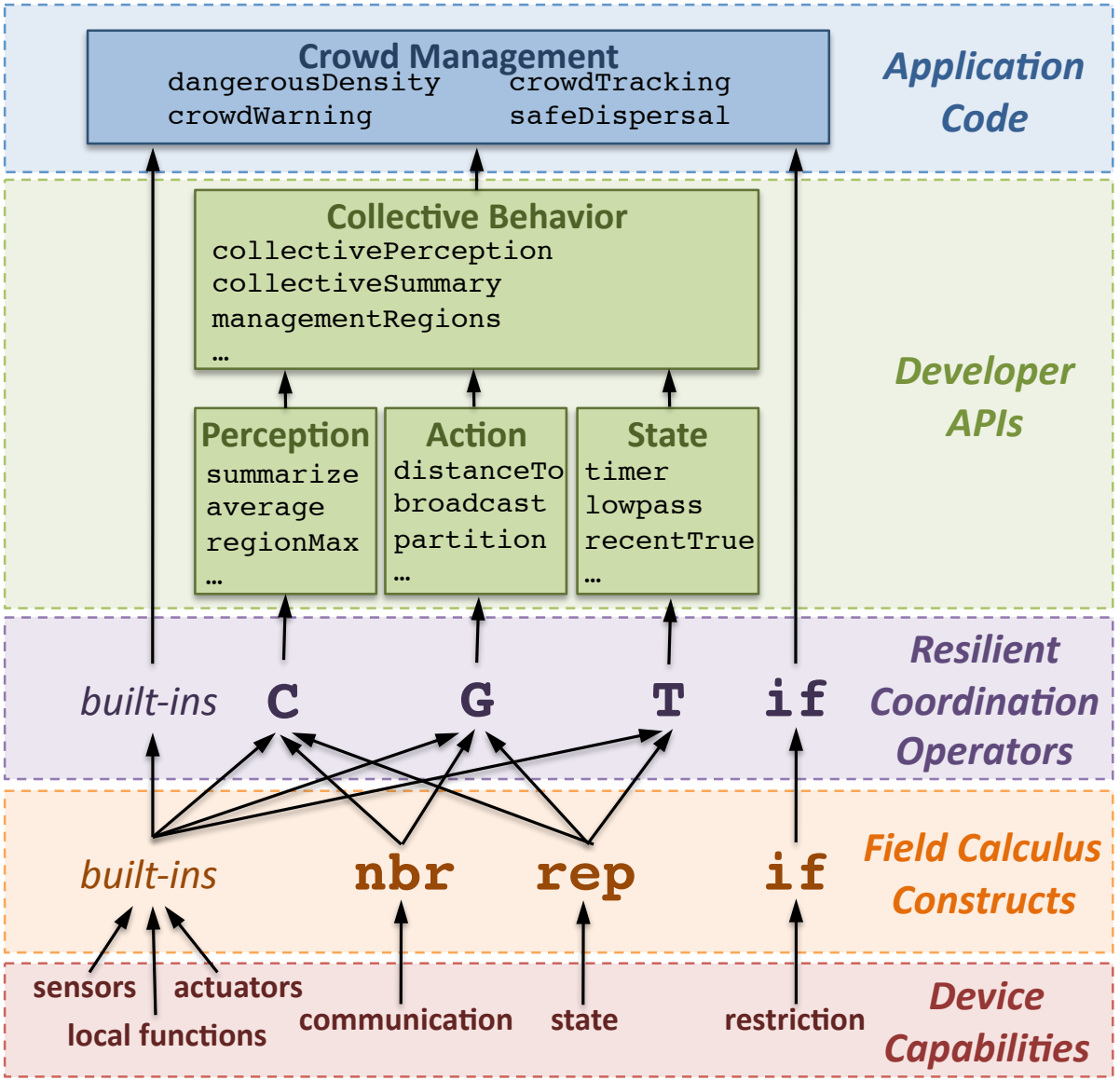
[Beal et al., 2012]

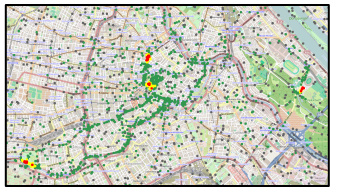
Summary: Aggregate Methodology

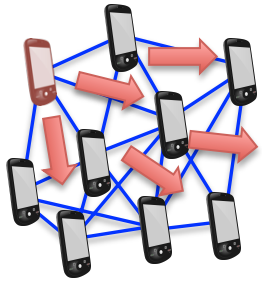
Raytheon BBN Technologies

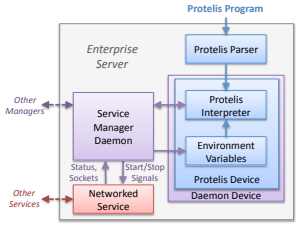












Summary

Raytheon
BBN Technologies

- Major technological trends are all driving towards a world filled with distributed systems
- Aggregate programming aims at rapid and reliable engineering of complex distributed systems
- Field calculus provides a universal theoretical foundation for aggregate programming
- Resilient systems design can be simplified by an emerging self-organization toolbox
- Functional composition allows modulation, predictable convergence

Lots of open problems…

Raytheon
BBN Technologies

- Many field calculus results may be more generally applicable to all distributed systems
- Extent of basis set needed for various domains
- What implicit propeties are needed/possible for building block algebras and libraries?
- Extension to mobile devices, ongoing changes
- Prediction of approximation quality – “Nyquist”
- Open environments and security

But we’ve already enough for many applications...

Bibliography

Raytheon BBN Technologies

- Beal J, Dulman S, Usbeck K, Viroli M, Correll N. Organizing the Aggregate: Languages for Spatial Computing. In: Mernik M, editor. Formal and Practical Aspects of Domain-Specific Languages: Recent Developments. IGI Global; 2013. p. 436–501.
- Viroli M, Damiani F, Beal J. A Calculus of Computational Fields. In: Canal C, Villari M, editors. Advances in Service-Oriented and Cloud Computing. vol. 393 of Communications in Computer and Information Sci. Springer Berlin Heidelberg; 2013. p. 114–128.
- Beal J, Bachrach J. Infrastructure for Engineered Emergence in Sensor/Actuator Networks. IEEE Intelligent Systems. 2006 March/April;21:10–19.
- Beal J, Viroli M, Damiani F. Towards a Unified Model of Spatial Computing. In: 7th Spatial Computing Workshop (SCW 2014). AAMAS 2014, Paris, France; 2014.
- Fernandez-Marquez J, Marzo Serugendo G, Montagna S, Viroli M, Arcos J. Description and composition of bio-inspired design patterns: a complete overview. Natural Computing. 2013;12(1): 43–67.
- Viroli M, Damiani F. A Calculus of Self-stabilising Computational Fields. In: 16th Conference on Coordination Languages and Models (Coordination 2014); 2014. p. 163–178.
- Beal J, Viroli M. Building blocks for aggregate programming of self-organising applications. In: Workshop on Foundations of Complex Adaptive Systems (FOCAS); 2014.
- Beal J. A Basis Set of Operators for Space-Time Computations. In: Spatial Computing Workshop; 2010.
- Jacob Beal, Mirko Viroli, Danilo Pianini, and Ferruccio Damiani. Scale-independent computations in situated networks. *under review.*

Acknowledgements

Raytheon
BBN Technologies

- Shane Clark
- Partha Pal
- Kyle Usbeck

- Soura Dasgupta
- Raghu Mudumbai
- Amy Kumar

- Mirko Viroli
- Danilo Pianini

- Ferruccio Damiani