An Architecture to Support Cognitive-Control of SDR Nodes

Karen Zita Haigh

khaigh@bbn.com

Roles for AI in Networking

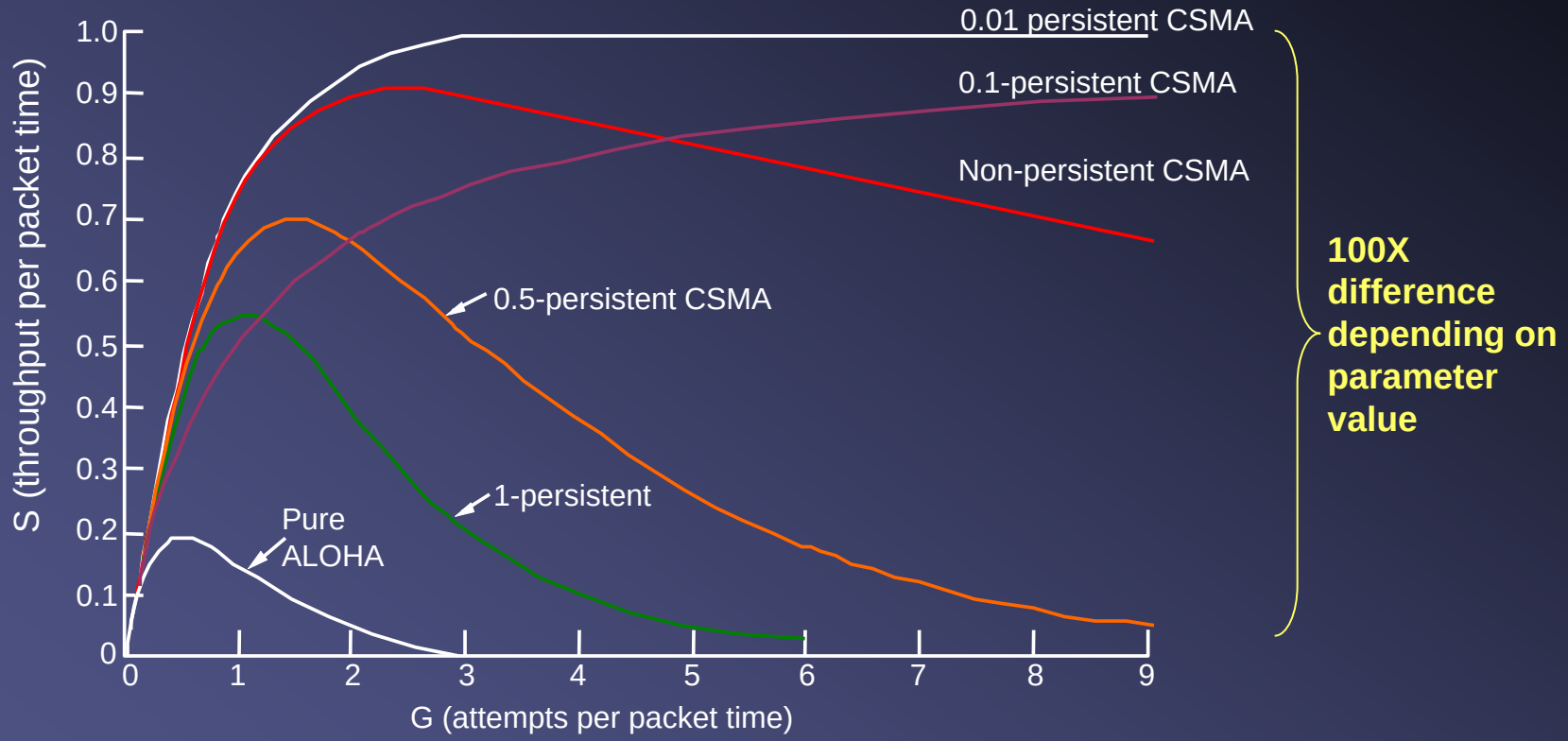
- Cyber Security
- Network Configuration (which modules to use)
- Network Control (which parameter settings to use)
- Policy Management
- Traffic Analysis

- Sensor fusion / situation assessment
- Planning
- Coordination
- Optimization
- Constraint reasoning
- Learning (Modelling)
  - Complex Domain
  - Dynamic Domain
    - Unpredictable by Experts

AI enables real-time, context-aware adaptivity

CSMA performance

This is the *same* protocol (Carrier Sense Multiple Access), with one simple parameter changed.



Which one should be the “default”?
Which one does the field commander really want?

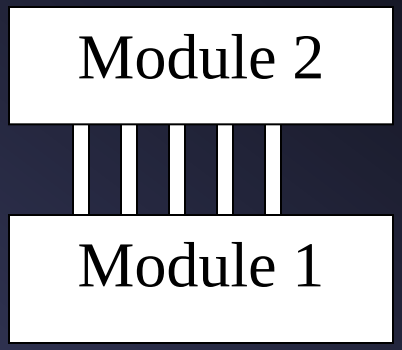
Network Control is ready for AI

- **Massive Scale**: ~600 observables and ~400 controllables *per node.*
- **Distributed:** each node must make its own decisions
- **Complex Domain:**
  - Complex & poorly understood interactions among parameters
  - Complex temporal feedback loops (at least 3: MAC/PHY, within node, across nodes); High-latency
- **Rapid decision cycle:** one second is a *long* time
- **Constrained:** Low-communication: cannot share all knowledge
- **Incomplete Observations:**
  - Partially-observable: some things can not be observed
  - Ambiguous observations: what caused the observed effect?

Human network engineers can't handle this complexity!

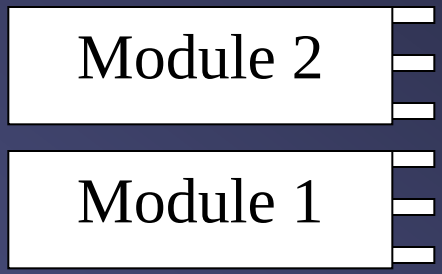
A Need for Restructuring

- SDR gives opportunity to create highly-adaptable systems, BUT
  - They usually require network experts to exploit the capabilities!
  - They usually rely on module APIs that are carefully designed to expose each parameter separately.
- This approach is not maintainable
  - e.g. as protocols are redesigned or new parameters are exposed.
- This approach is not amenable to real-time cognitive control
  - Hard to upgrade
  - Conflicts between module & AI

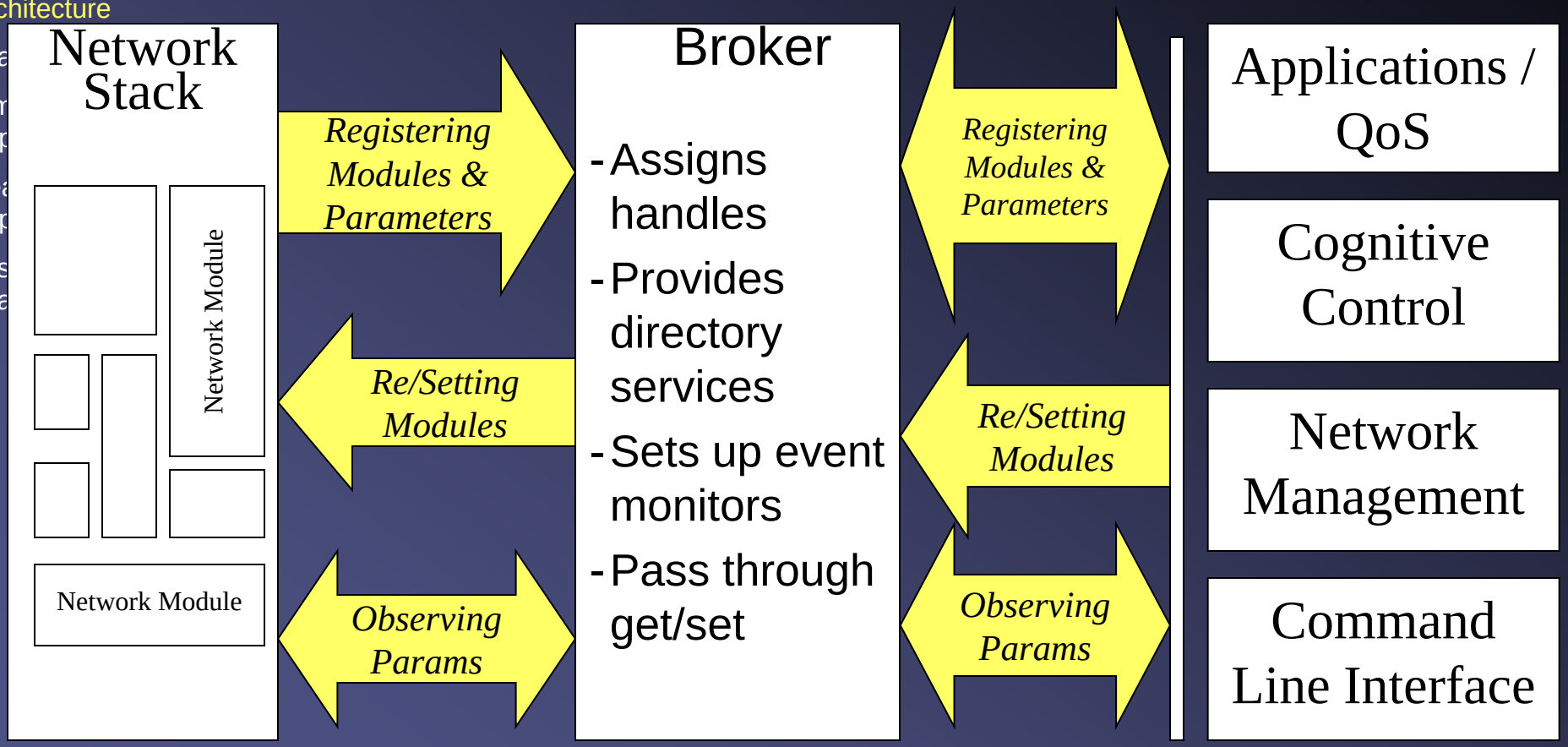


A Need for Restructuring

- We need one consistent, generic, interface for all modules to expose their parameters and dependencies.



A Generic Network Architecture



exposeParameter(*parameter_name*, *parameter_properties*)
setValue(*parameter_handle*, *parameter_value*)
getValue(*parameter_handle*)

Benefits of a Generic Architecture

- It supports network architecture design & maintenance
  - Solves the *nxm* problem (upgrades or replacements of network modules)
- It doesn't restrict the form of cognition
  - Open to just about any form of cognition you can imagine
  - Supports *multiple* forms of cognition on each node
  - Supports *different* forms across nodes

A problem formulation:

Distributed Optimization

Honeywell

Problem Formulation (1)

- Consider a MANET with **N** heterogeneous nodes
- Each node ***i*** has a set of m_i control parameters x_i
  - Parameters that control the behaviour of the protocols
- Each node ***i*** has a set of n_i observable parameters y_i
  - Context that can be observed
- Note, there may be unobservable parameters **z**.

Problem Formulation (2)

- Associated with the MANET is a scalar performance measure **J(t)** that characterizes global network performance
  - Throughput, Latency, User needs, Mission, etc
  - **J(t)** = f(controllables, observables, unobservables), for all nodes **N**, over all previous time **0,…,t**
    - Note: **O(N** x 1000 x **t)** elements to calculate **J!**

- Goal: Optimize **J(t)**, despite
  - No exact expression for **J** (notably unobservables)
  - Distributed: each node ***i*** determines its own control values x_i
  - Over time
  - Keep overhead low (i.e. use as few observables from other nodes as possible; keep coordinated)

ORACLE (Machine Learner)

ORACLE: Optimizing Rapidly Adaptive Configuration Learning Engine

- ORACLE builds a model of the performance surface based on empirical data
  - Each node ***i*** builds a model of **J**
- This is hard because
  - Extremely large search space (**N** x 1000 x **t**)
  - Complex temporal feedback issues
- We have no exact expression for **J**

- We simplify by
  - Using *only* local observables and controllables
    - Assumes that behaviour of other nodes will be observed locally, e.g. if a neighbour increases data rate, node will see increased congestion
  - Memory-less (i.e. no time; use most recent measurements)
    - Assumes that prior data has affected the model of the performance surface

Modelling performance

- Each node learns how to relate its *own* observables and controllables to global network performance
  - Permits but does not require inter-node communication
- Each node may have a *unique* model (i.e. different from other nodes)

- **i**'s estimate of $\hat{\mathbf{J}}(t) = f_i (\mathbf{x_i}(t), \mathbf{y_i}(t-1))$

Control parameters at time t

Observable parameters from time t-1

ORACLE approach

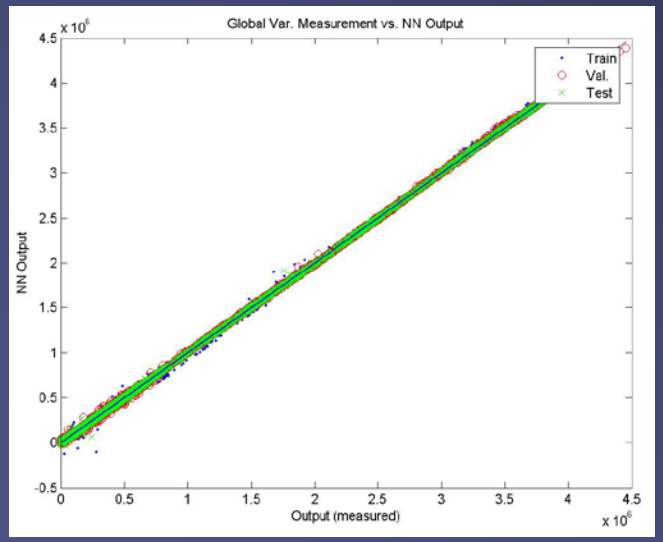
- Optimizing Rapidly Adaptive Configuration Learning Engine (ORACLE)
- Unique hybrid approach: Combines Analytical Network models and Machine Learning
- Analytical Models: a priori models of network behaviour
  - Capture useful general principles
  - But are incomplete, incorrect, and static
- Machine Learning: empirical models built from experience
  - Capture actual operating conditions
  - But poorly transfer knowledge to new domains or objective functions

Simulated Experiments

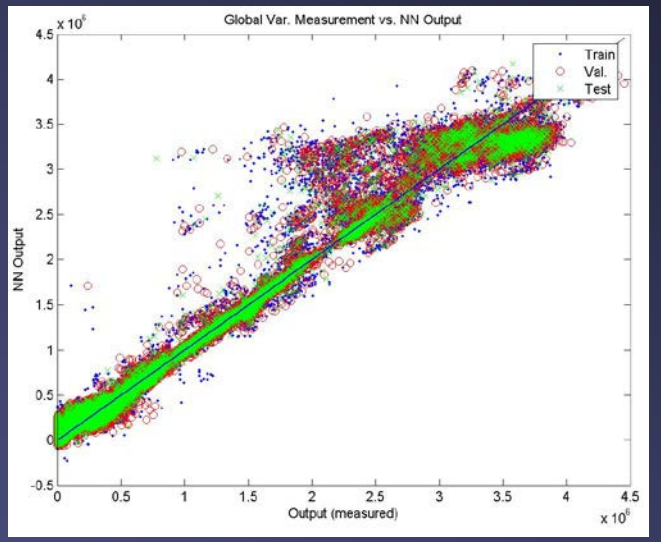
- Simplified MANET scenario, 4-stage battle
- Control settings:
  - Network layer: 1,4,8 second Hello Interval
  - MAC: 2,4,8 max retransmissions
  - PHY: 1,2,11 MBps data rate (transmit power levels are implicitly controlled in 802.11b)
- Training data:
  - 27 homogeneous & 90 heterogeneous cases
  - Local observations at each second at each node
  - Train Artificial Neural Network (ANN), one per node
- Testing:
  - One test run, set control parameters once per second

Simulated Experiments

Phase	Mobility	Data
1: Deploy	No motion	1024-bye packets, constant bit-rate
2: Shape	Slow (5 min)	100-byte packets, CBR
3: Decisive Ops	Fast (1 min)	100-byte packets, CBR
4: Consolidate	No motion	1024 byte packets, CBR



Model accuracy for stationary node



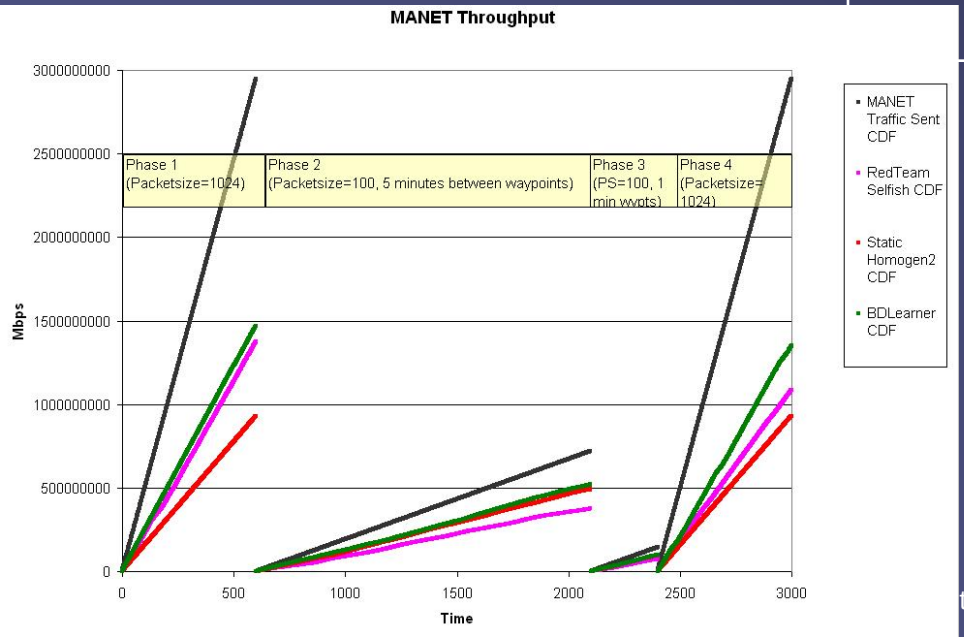
Model accuracy for mobile node

Does Learning work?

Experiment #1 – Learner compared to Standard Approaches

- Red Team (human Expert)
- Best static homogeneous setting
- Learned

Phase End	Learner	Red Team	Static Homog.
1	1,470 MB	1,376 MB 94%	929 MB 63%
2	520 MB	375 MB 72%	491 MB 94%
3	96 MB	72 MB 75%	97 MB 101%
4	1,350 MB	1,086 MB 80%	9320 MB 69%

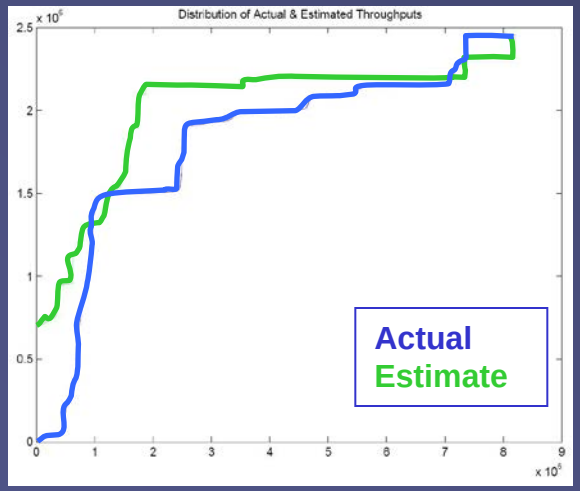


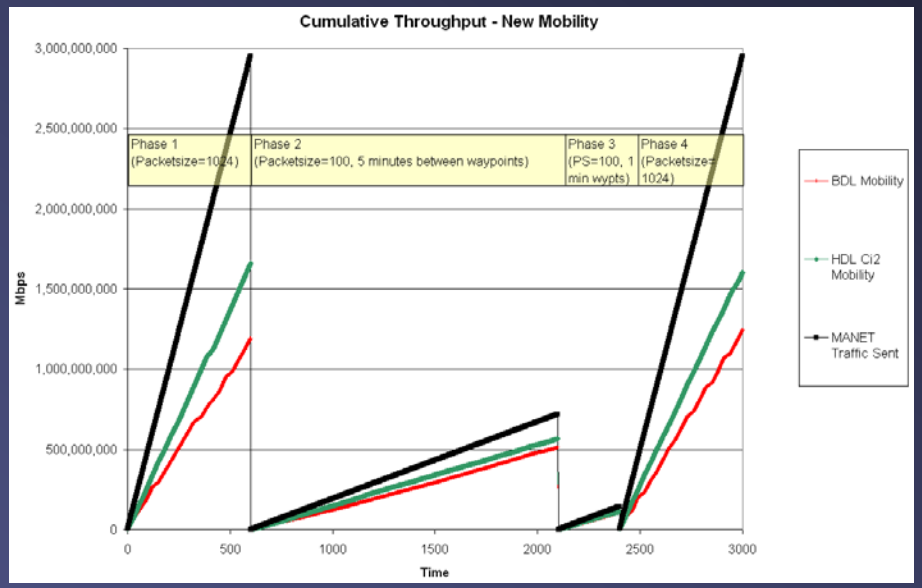
Can analytical models help?

Experiment #2A – Knowledge Transfer

- Change Mobility Patterns
  - Training mobility patterns are changed before the test
- Learners
  - Basic (strictly learned)
  - Hybrid (adds analytical estimate of throughput)

Phase End	Basic Learner	Hybrid Learner	Improvement
1	1,192 MB	1,660 MB	139%
2	515 MB	567 MB	110%
3	107 MB	113 MB	105%
4	1,246 MB	1,604 MB	129%
Total	3,061 MB	2,944 MB	129%



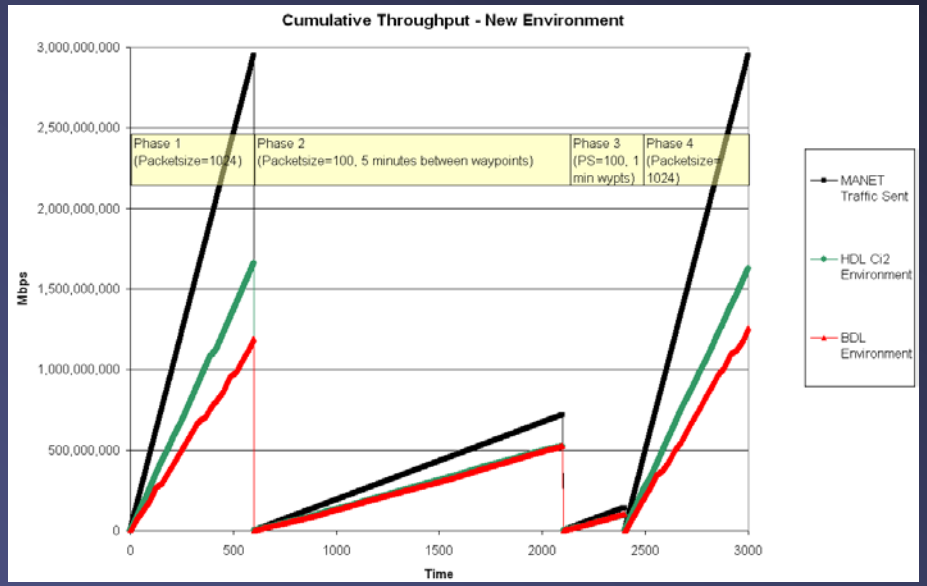


Can analytical models help?

Experiment #2B – Knowledge Transfer

- Change Communications environment
- Learners
  - Basic (strictly learned)
  - Hybrid (adds analytical estimate of throughput)

Phase End	Basic Learner	Hybrid Learner	Improvement
1	1,192 MB	1,660 MB	139%
2	528 MB	525 MB	99%
3	103 MB	97 MB	93%
4	1,260 MB	1,627 MB	129%
Total	3,083 MB	3,909 MB	127%



A real-world example:

Adaptive Dynamic Radio Open-source Intelligent Team (ADROIT)

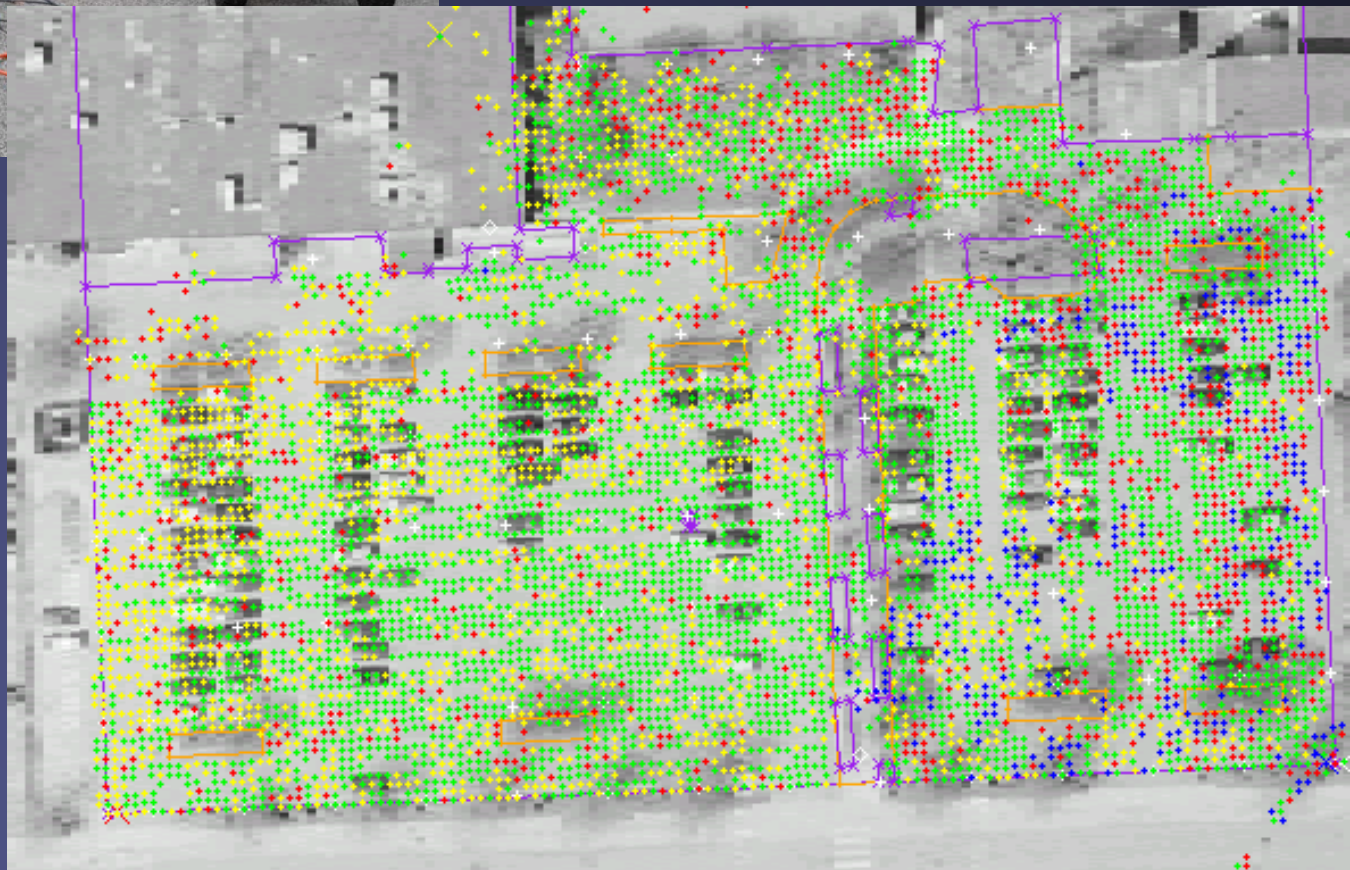
BBN, UKansas, UCLA, MIT

ADROIT's mission

- DARPA project: ACERT, 2006
- Create cognitive radio teams with both *real-time composability* of the stack and *cognitive control* of the network.

- Recognize that the situation has changed
- Anticipates changes in networking needs
- Adapts the network, in real-time, for improved performance
  - Real-time composability of the stack
  - Real-time Control of parameters
  - On one node or across the network

Experimental Testbed



Maximize % of shared map of the environment

Experiment Description

- Maximize % of shared map of the environment
- **Goal:** Choose Strategy to maximize expected outcome given Conditions.
  - Each node chooses independently, so strategies must be interoperable
- Measure conditions
  - signal strength from other nodes
  - location of each node

Strategies:

- 2 binary strategy choices for 4 strategies

1. How to send fills to nodes without data?
   - multicast, unicast
2. When to send fills?
   - always
   - if we are farthest (and data is not ours), refrain from sending

Experimental Results

Training Run:

- In first run nodes learn about environment
- Train neural nets with (C,S)➔P tuples
  - Every 5s, measure and record progress conditions, strategy
  - Observations are local, so each node has different model!

Real-time learning run:

- In second run, nodes adapt behaviour to perform better.
- Adapt each minute by changing strategy according to current conditions

Real-time cognitive control of a real-world wireless network

Observations from Learning

System performed better with learning

- Selected configurations explainable but not predictable
  - Farthest-refraining was usually better
    - congestion, not loss dominated
  - Unicast/Multicast was far more complex
    - close: unicast wins (high data rates)
    - medium: multicast wins (sharing gain)
    - far: unicast wins (reliability)

BBN TECHNOLOGIES

Intro

Architecture

Learning

Simulated Experiments

Real World Experiments

Lessons Learned

Overcoming Cultural Differences to Get a Good Design

Cultural Issues: But why?

- Benefits and scope of cross-layer design:
  - More than 2 layers!
  - More than 2-3 parameters per layer

➢ Drill-down walkthroughs highlighted benefits to networking folks; explained restrictions to AI folks

➢ Simulation results for specific scenarios demonstrated the power

- Traditional network design includes adaptation
  - But this works against cognition: it is hard to manage *global* scope
  - AI people want to control everything
  - But network module may be better at doing something focussed

➢ Design must include *constraining* how a protocol adapts

Cultural Issues: But how?

- Reliance on centralized Broker:
  - Networking folks don't like the single bottleneck
  - ➢ Design must have fail-safe default operation
- Asynchrony and Threading:
  - AI people tend to like blocking calls.
    - e.g. to ensure that everything is consistent
  - Networking folks outright rejected it.
  - ➢ Design must include reporting and alerting

Cultural Issues: But it'll break!?!

- Relinquishing control outside the stack:
  - Outside controller making decisions scares networking folks
  - AI folks say "give me everything & I'll solve your problem"

➢ Architecture includes "failsafe" mechanisms to limit both sides

- Heterogenous and *non-interoperable* nodes
  - Networks usually have homogeneous configurations to maintain communications
  - AI likes heterogeneity because of the benefit
    - But always assumes safe communications!

➢ "Orderwire" bootstrap channel as backup

Cultural Issues: New horizons?

- Capability Boundaries
  - Traditional Networking has very clear boundary between “network” and “application”
  - Generic architecture blurs that boundary
    - AI folks like the benefit
    - Networking folks have concerns about complexity

➢ Removing this conceptual restriction will result in interesting and significant new ideas.

Conclusion

- AI in networks is a Good Thing.
- Traditional network architectures do not support cognition
  - Hardware is doing that now (SDR), but the software needs to do the same thing
- To leverage the power of cognitive networking, both AI folks & Networking folks need to recognize and adapt

Papers & Resources

- K. Z. Haigh, S. Varadarajan, C. Y. Tang, "Automatic Learning-based MANET Cross-Layer Parameter Configuration," in *IEEE Workshop on Wireless Ad hoc and Sensor Networks (WWASN)*, Lisbon, Portugal 2006.
- K. Z. Haigh, O. Olofinboba, C. Y. Tang, "Designing an Implementable User-Oriented Objective Function for MANETs," in *IEEE International Conference On Networking, Sensing and Control*, London, U.K. April 15-17, 2007.
- G. D. Troxel et al. "Enabling open-source cognitively-controlled collaboration among software-defined radio nodes." *Computer Networks,* 52(4):898-911, March 2008.
- G. D. Troxel et al, "Cognitive Adaptation for Teams in ADROIT," in *IEEE Global Communications Conference*, Nov 2007, Washington, DC. **Invited**.
- Getting the ADROIT Code (Including the Broker)
  - https://acert.ir.bbn.com/
  - checkout instructions
  - GNU Radio changes are in main GNU Radio repository

ADROIT Team

BBN Technologies:

- Greg Troxel (PI), Isidro Castineyra (PM)
- *AI*: Karen Haigh, Talib Hussain
- *Networking*: Steve Boswell, Armando Caro, Alex Colvin, Yarom Gabay, Nick Goffee, Vikas Kawadia, David Lapsley, Janet Leblond, Carl Livadas, Alberto Medina, Joanne Mikkelson, Craig Partridge, Vivek Raghunathan, Ram Ramanathan, Paul Rubel, Cesar Santivanez, Dan Sumorok, Bob Vincent, David Wiggins
- Eric Blossom (GNU Radio consultant)

University of Kansas:

- Gary Minden, Joe Evans

MIT: Robert Morris, Hari Balakrishnan

UCLA: Mani Srivastava