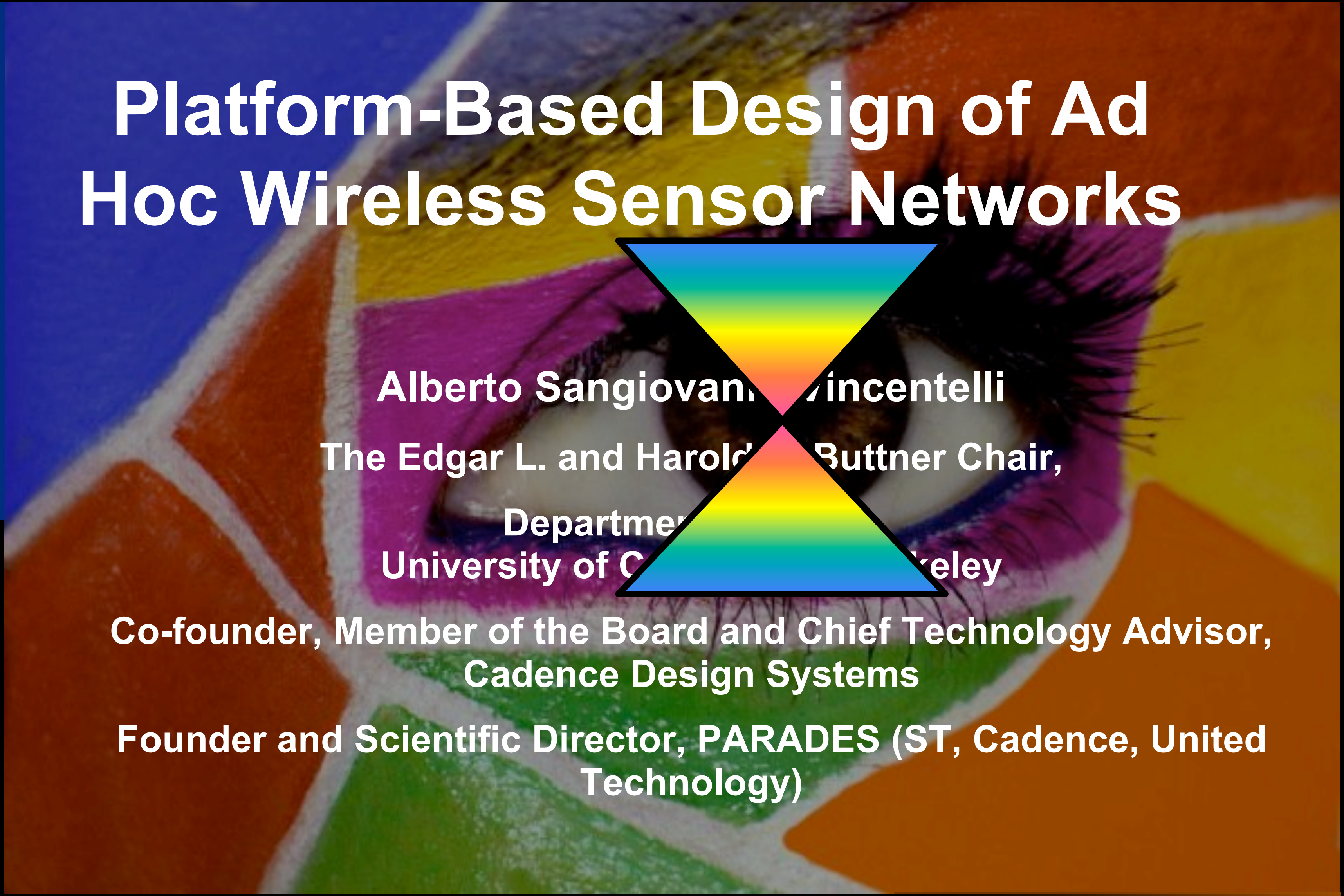


Platform-Based Design of Ad Hoc Wireless Sensor Networks

Alberto Sangiovan incentelli

The Edgar L. and Harold Buttner Chair,

Departme
University of C keley

Co-founder, Member of the Board and Chief Technology Advisor, Cadence Design Systems

Founder and Scientific Director, PARADES (ST, Cadence, United Technology)

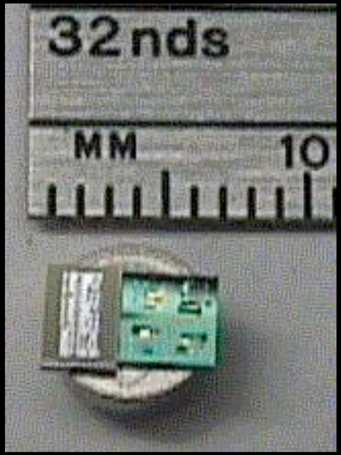
Outline

- Wireless Sensor Networks and their Evolution
- Platform-based Design for AWSN
  - Sensor Network Service Platform
  - Sensor Network Implementation Platform
- Design Flow
  - Rialto: specification capture (SNPS)
  - Genesis: protocol synthesis (SNIP)

Wireless Sensor Networks

The use of wireless networks of embedded computers "could well dwarf previous milestones in the information revolution" - National Research Council Report: Embedded, Everywhere", 2001.

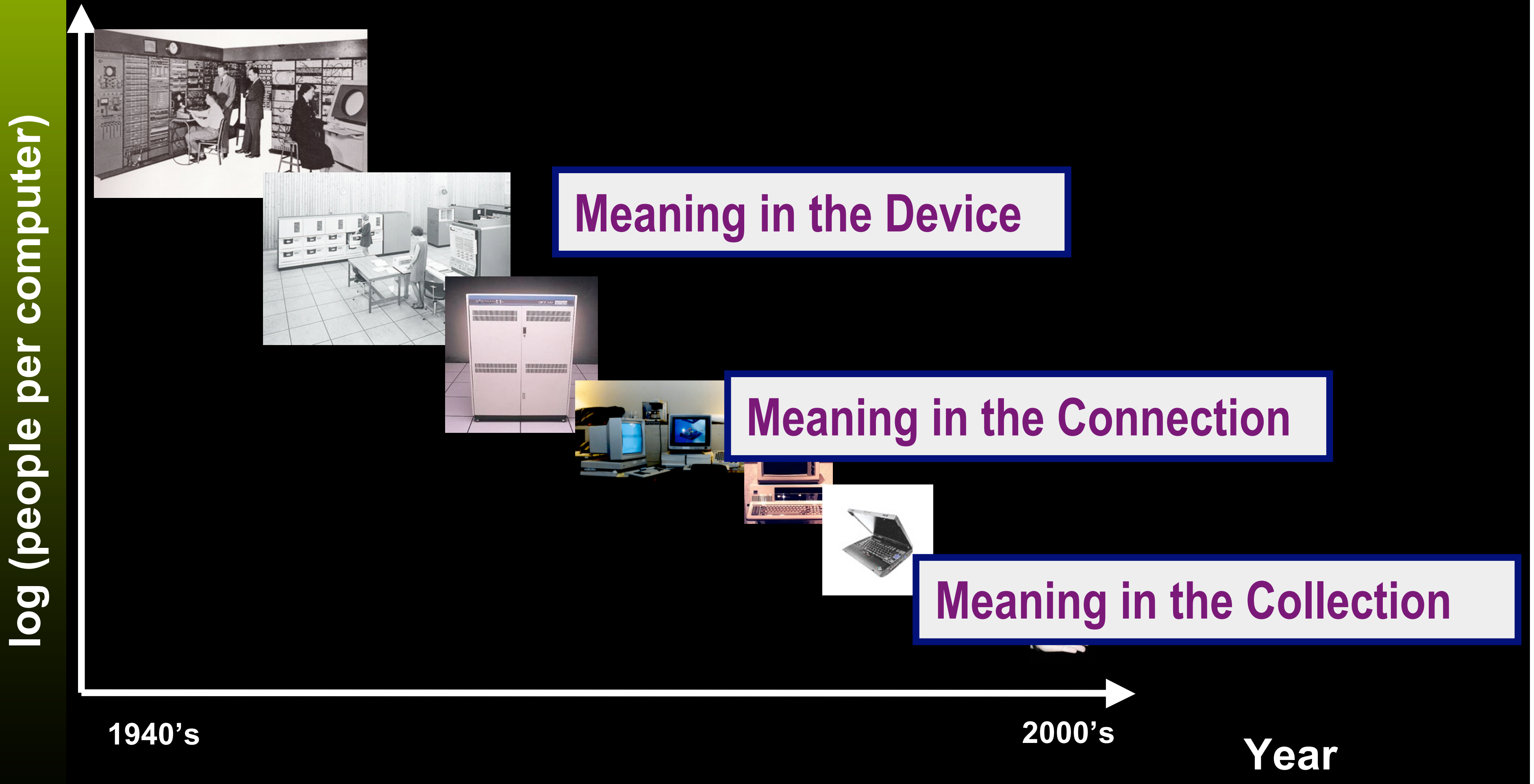
Berkeley Dust Mote[1]



Berkeley Mote[1]

[1]From Pister *et al.*, *Berkeley Smart Dust Project*

Bell's Law: A New Computer Class Every 10 Years


log (people per computer)
Meaning in the Device
Meaning in the Connection
Meaning in the Collection
1940's
2000's
Year

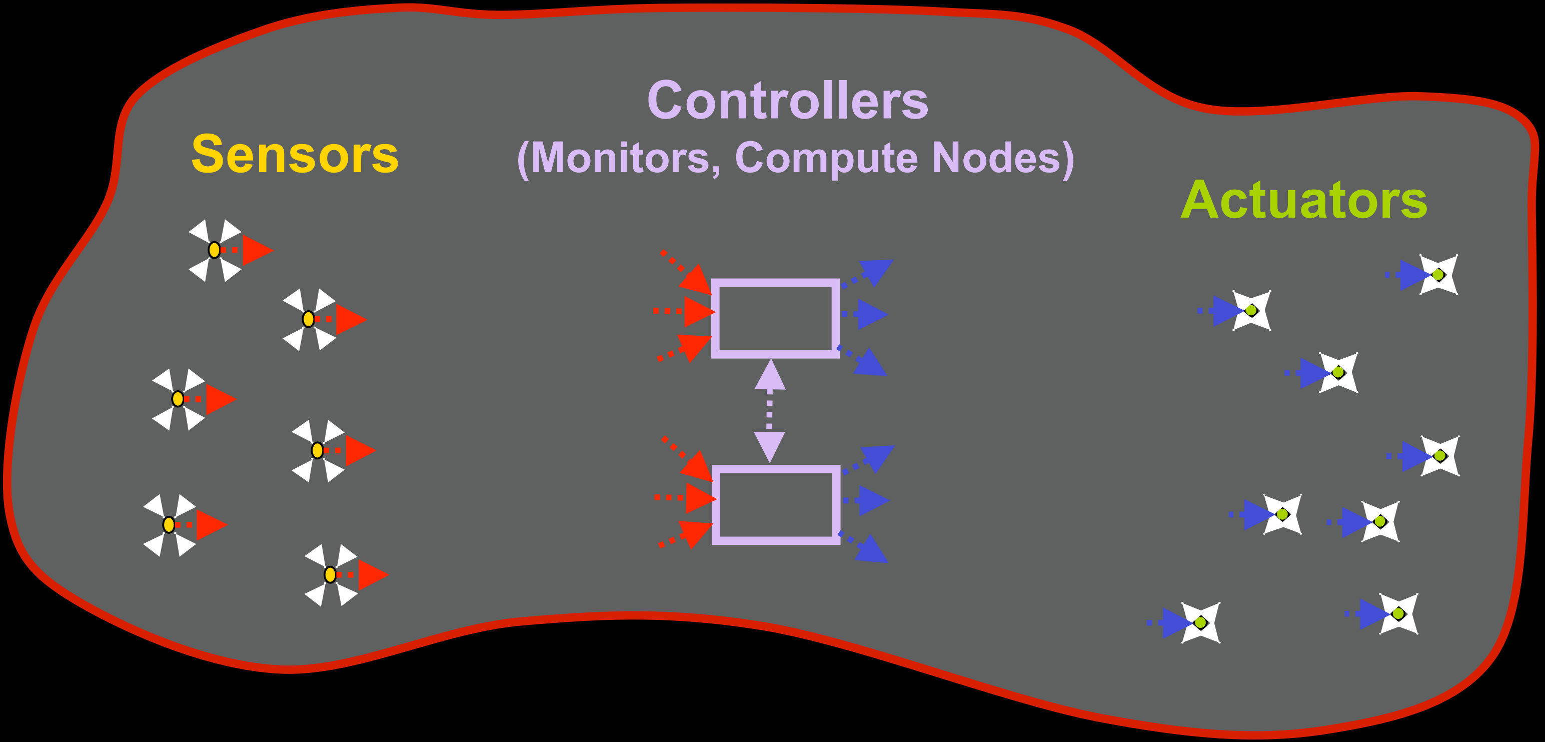
Courtesy: R. Newton

Ambient Intelligence

- An environment where technology is embedded, hidden in the background
- An environment that is sensitive, adaptive, and responsive to the presence of people and objects
- An environment that augments activities through smart non-explicit assistance
- An environment that preserves security, privacy and trustworthiness while utilizing information when needed and appropriate

Courtesy: Fred Boekhorst, Philips, ISSCC02

Wireless Sensor and Actuator Networks as a First Incarnation



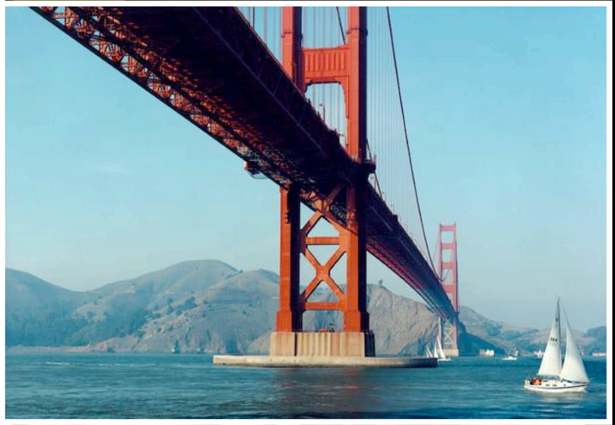
A collection of cooperating algorithms (controllers) designed to achieve a set of common goals, aided by interactions with the environment through **distributed** measurements (sensors) and actions (actuators).

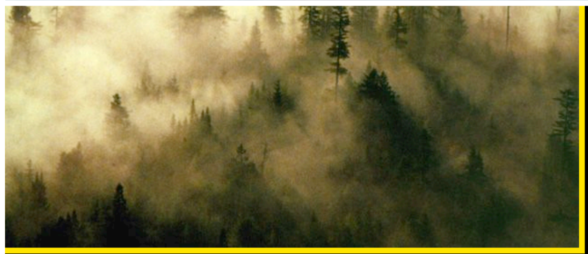
$150 millions sales in 2003

$7 billion in sales in 2010

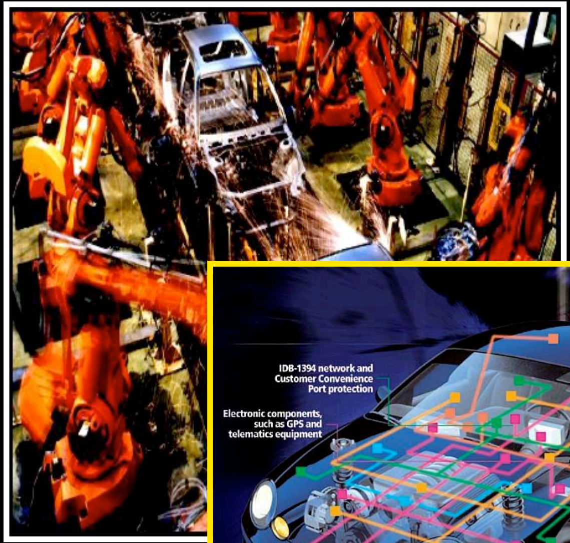
Creating a Whole New World of Applications

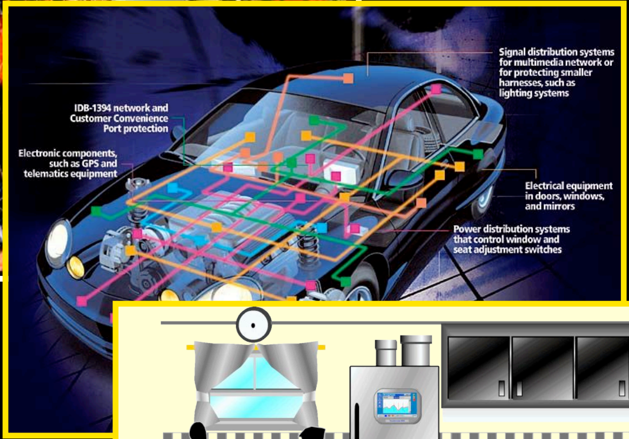
From Monitoring





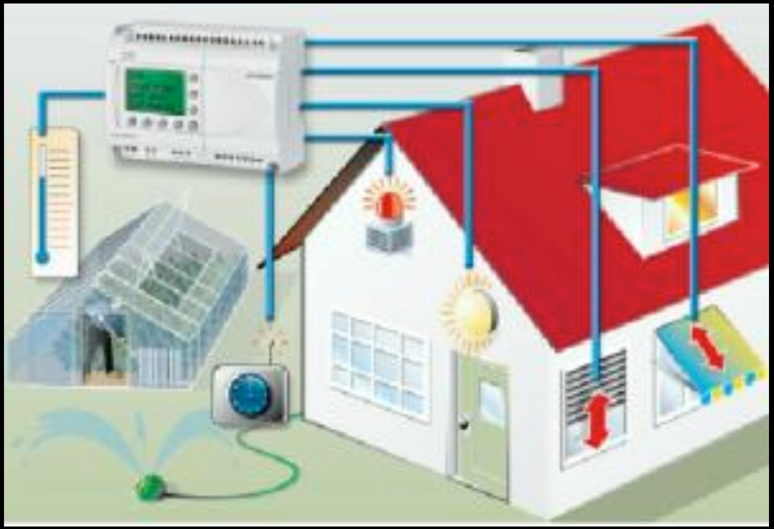
To Automation



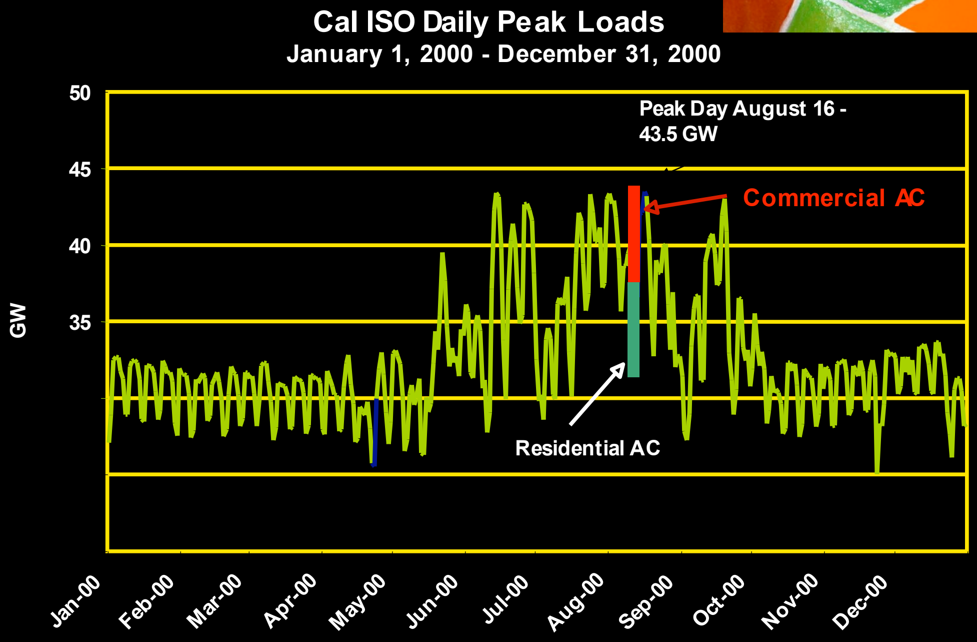

Signal distribution systems for multimedia network or for protecting smaller harnesses, such as lighting systems
IDB-1394 network and Customer Convenience Port protection
Electronic components, such as GPS and telematics equipment
Electrical equipment in doors, windows, and mirrors
Power distribution systems that control window and seat adjustment switches

The “Smart” Home — A Prime Target

- Security
- Environment control
- Energy management
- Object tracking/inventory
- Advanced user interfaces
- Sense of presence and space



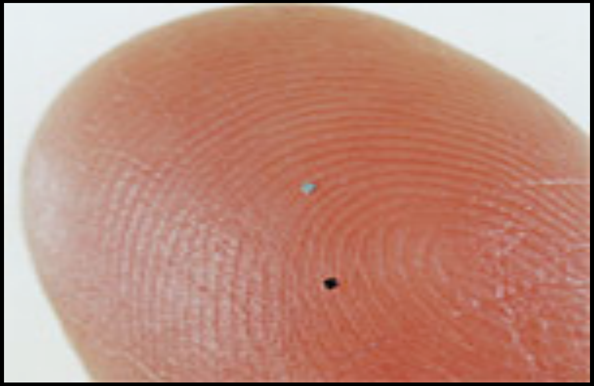
Energy Management and Conservation as the Initial Driver



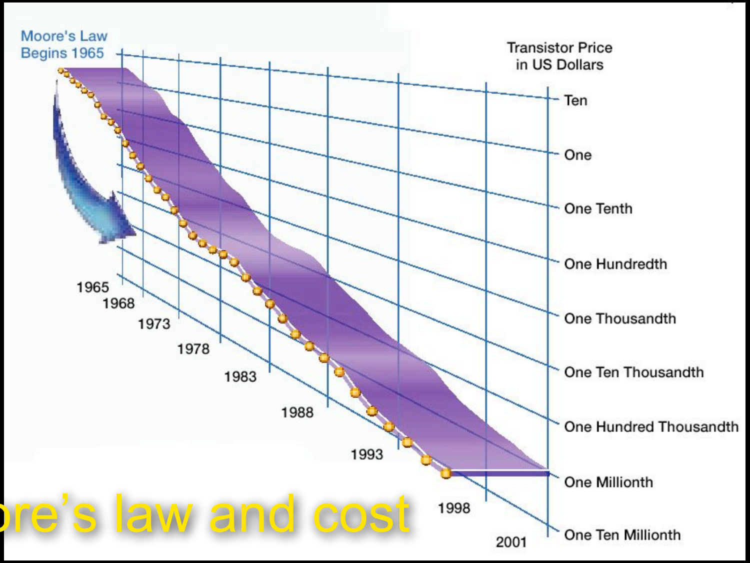
Demand response:
Make energy prices dependent upon time-of-use

- Advanced thermostats operate on required level of comfort, energy cost, weather forecast and distributed measurements to offload peak times
- Appliances are energy and cost aware

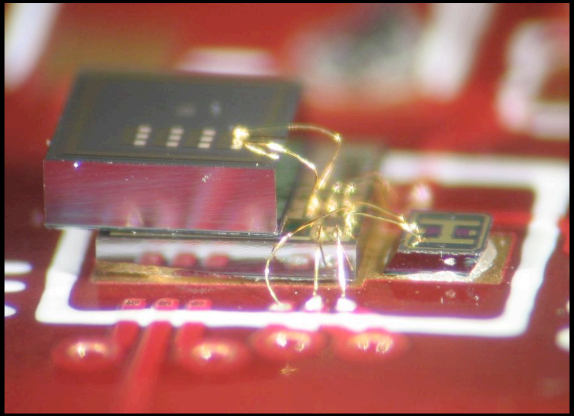
Enabled by Combined Technology Advancements



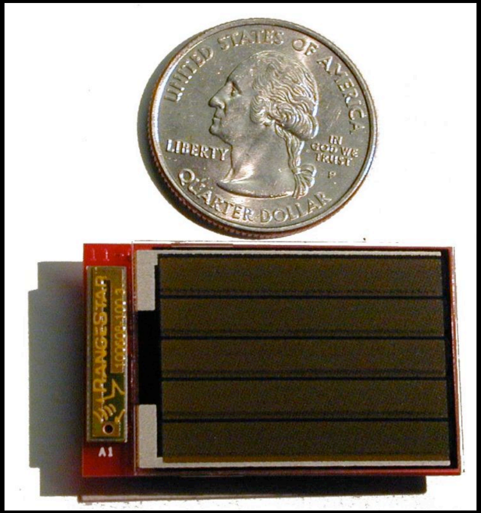
Moore's law and size



Moore's law and cost



True system integration



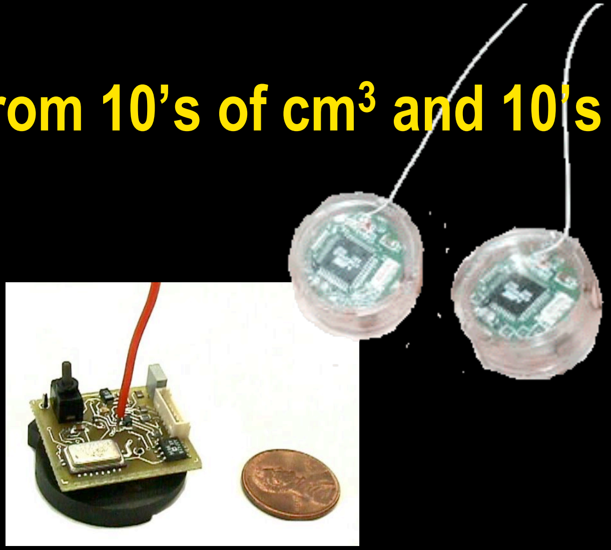
Ubiquitous wireless as the glue

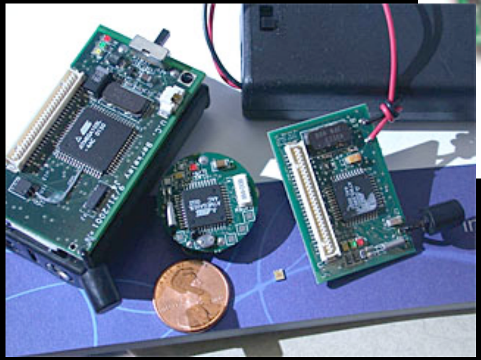
Challenges in Wireless Sensor Networks

- Power, Cost, Size (Disappearing Electronics)
- Reliability
- Portability, Scalability and Configurability
- *Security and Privacy*

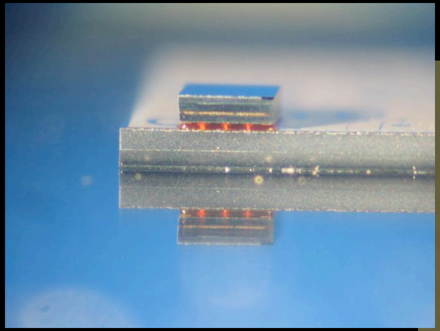
How to Make Electronics Truly Disappear?

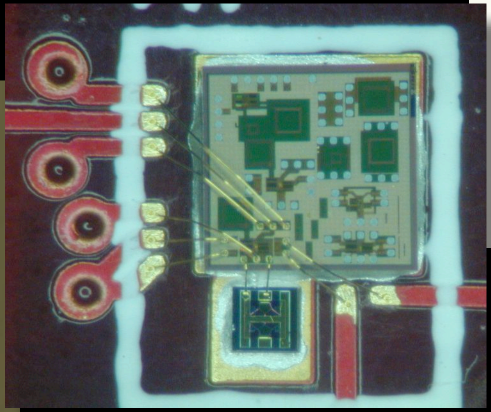
From 10's of cm^3 and 10's to 100's of mW

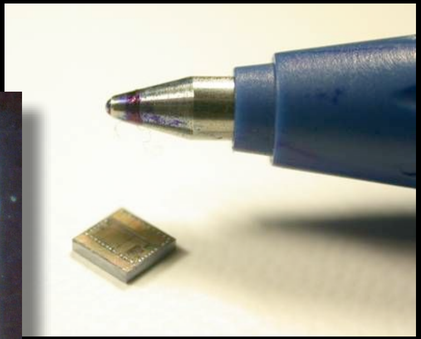




To 10's of mm^3 and 10's of μW

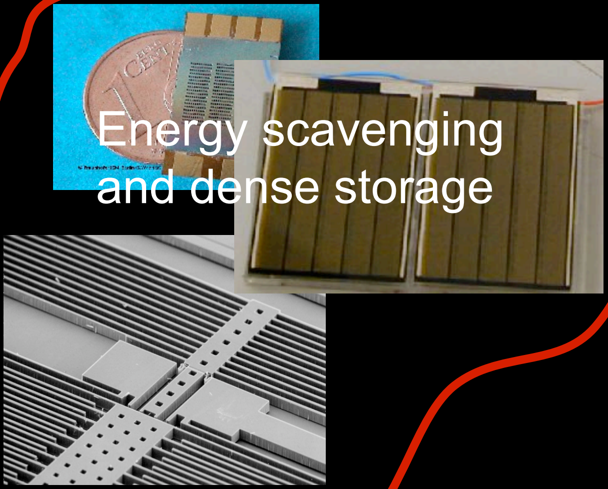






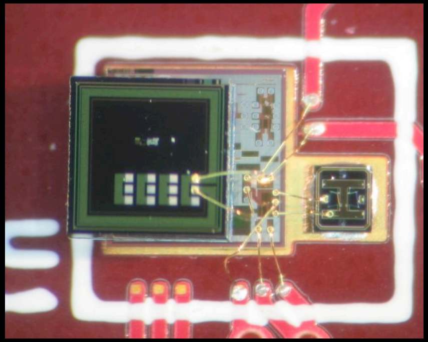
Copyright: A. Sangiovanni-Vincentelli

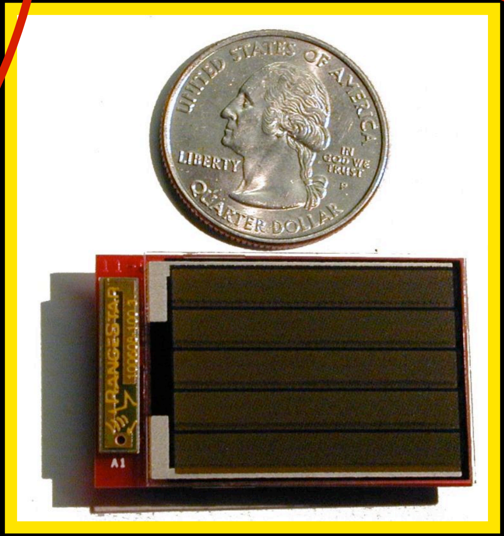
Adopt Non-Orthodox Technologies and Approaches



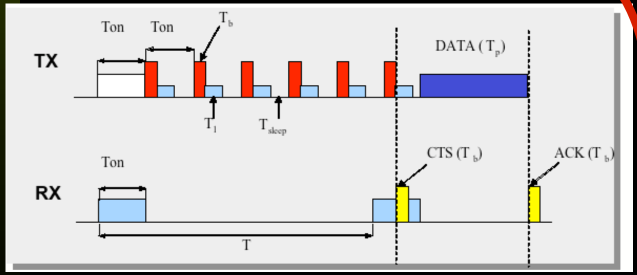
Energy scavenging and dense storage

Simple radio's with minimal external components

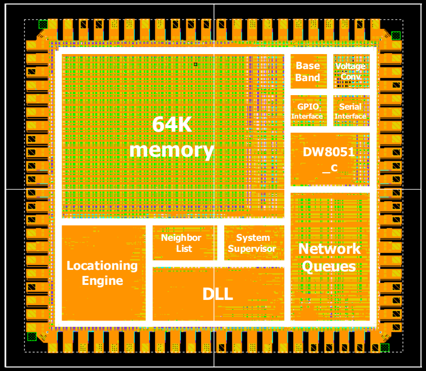




A 100 mW self-contained sensor node



Protocols that maximize standby time



Ultra-low power processors with low standby power

Reliability: A Serious Threat to Wireless Sensor Networks

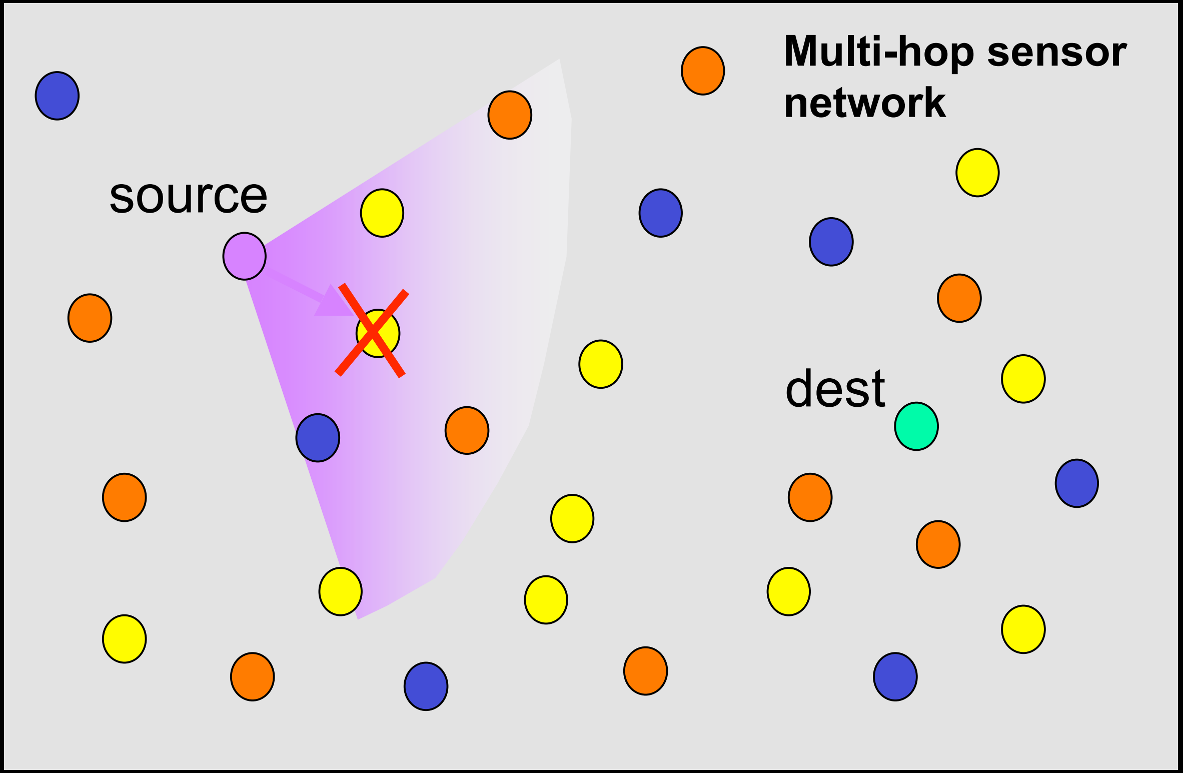
Unreliability is intrinsic to the disappearing electronics concept.

- Nodes may appear at will, may move, may fail and (temporarily) run our of energy
- Problem aggravated by cost, power and size constraints

The wrong answer: over-design

The right answer: exploit nature of Ambient Intelligence

Reliability through Redundancy

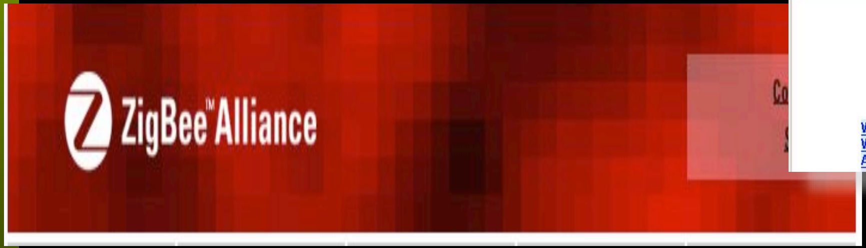


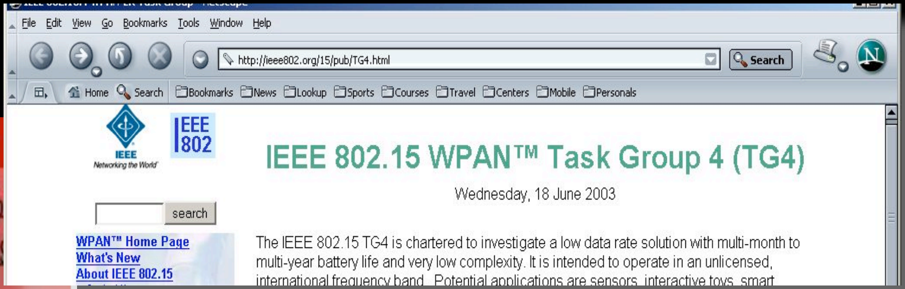
Opportunistic routing: choose any node that is available and goes in the right direction.
Also reduces energy and latency.

Redundancy is a core concept of Sensor Networks,
providing high quality reliable service with cheap unreliable components.

Scalability, Portability and Configurability

A plethora of implementation strategies emerging at all layers, some of them being translated into standards

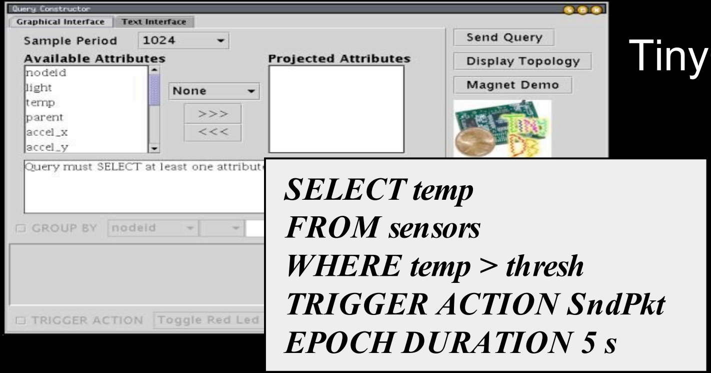



File Edit View Go Bookmarks Tools Window Help
http://ieee802.org/15/pub/TG4.html
Search
Home Search Bookmarks News Lookup Sports Courses Travel Centers Mobile Personals
IEEE 802
search
WPAN™ Home Page
What's New
About IEEE 802.15

IEEE 802.15 WPAN™ Task Group 4 (TG4)

Wednesday, 18 June 2003

The IEEE 802.15 TG4 is chartered to investigate a low data rate solution with multi-month to multi-year battery life and very low complexity. It is intended to operate in an unlicensed, international frequency band. Potential applications are sensors, interactive toys, smart

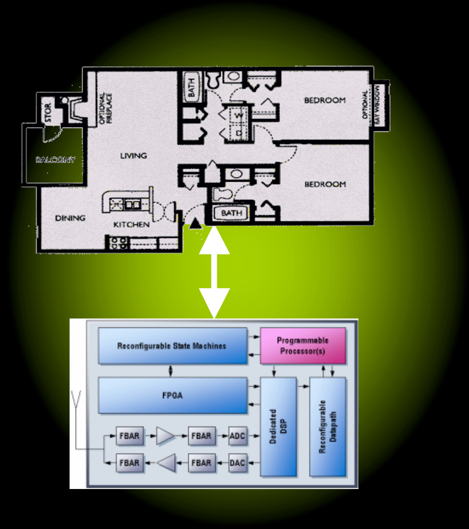


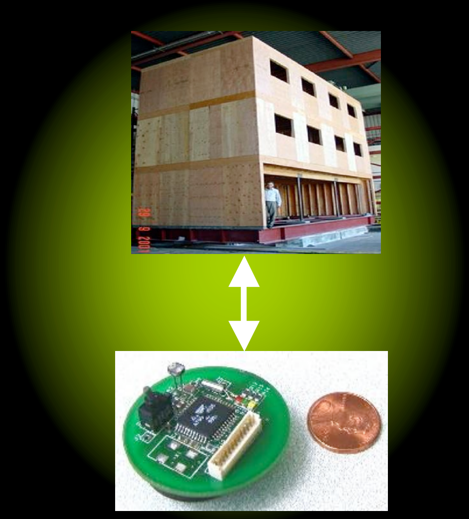
SELECT temp
FROM sensors
WHERE temp > thresh
TRIGGER ACTION SndPkt
EPOCH DURATION 5 s

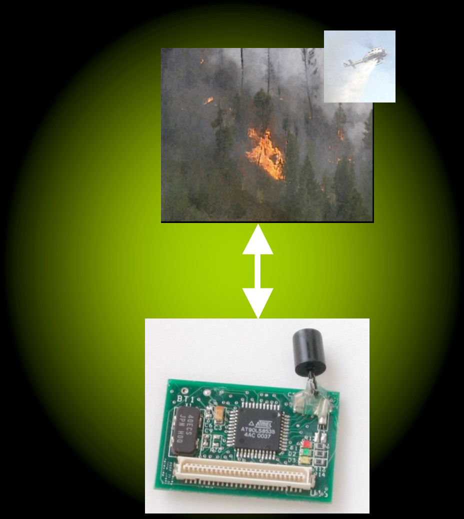
TinyOS/TinyDB

- Bottom-up definition without perspective on interoperability and portability
- Mostly "stovepipe" solutions
- Little reflection on how this translates into applications

Applications and Platforms Interoperability

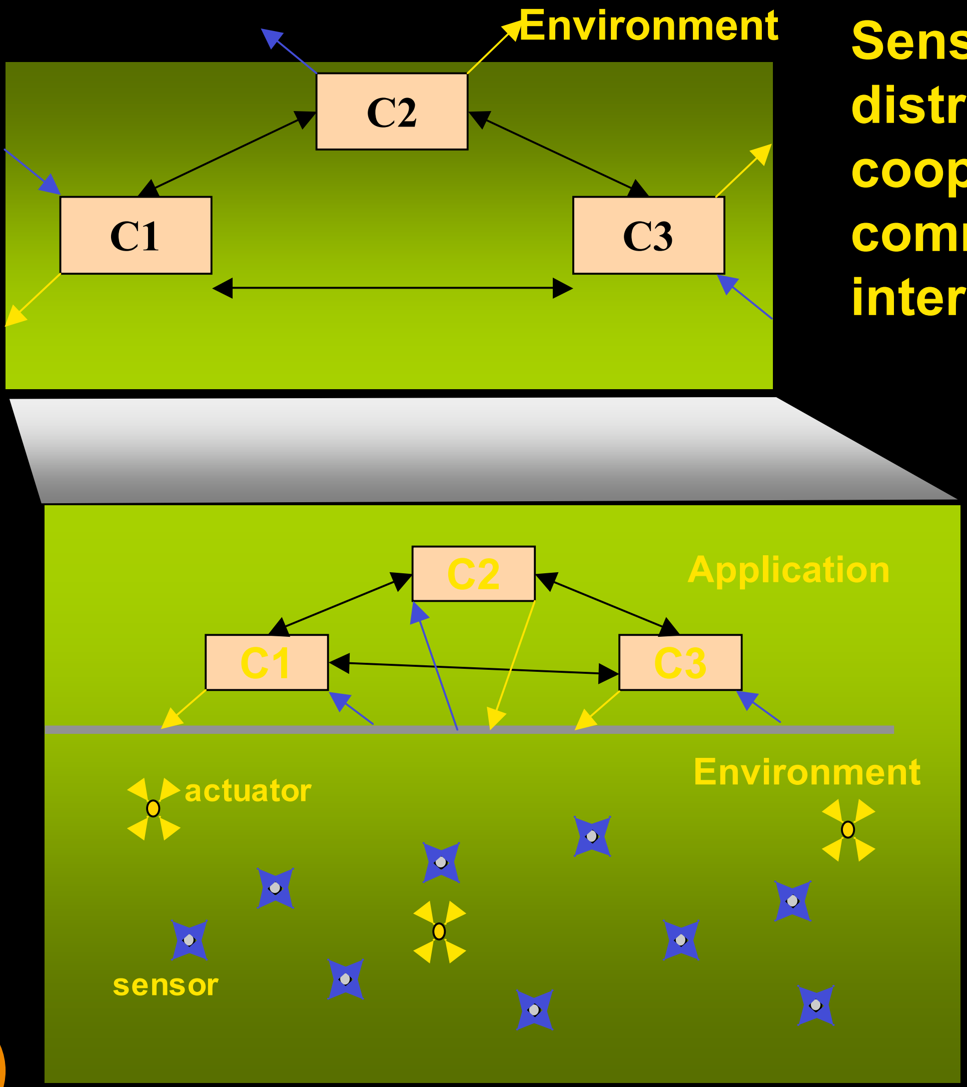






- Applications ***bound to specific*** implementation platforms
- Need ***interoperability*** between applications and between implementation platforms
- Need to ***hide*** implementation details from application programmers

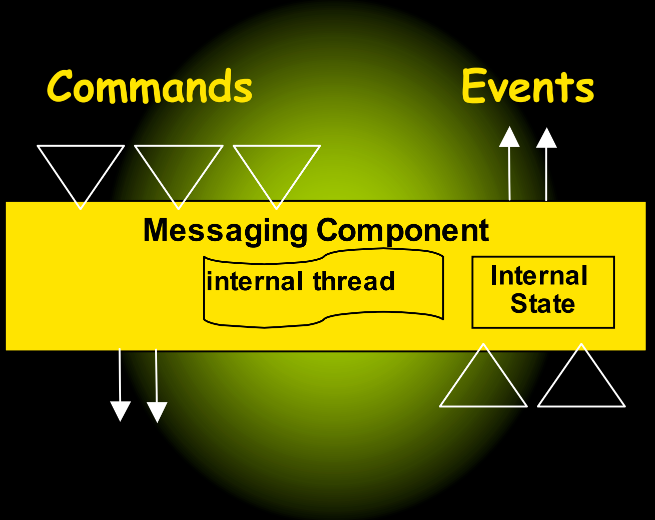
The Only Real Option: Raising the Abstraction Level!

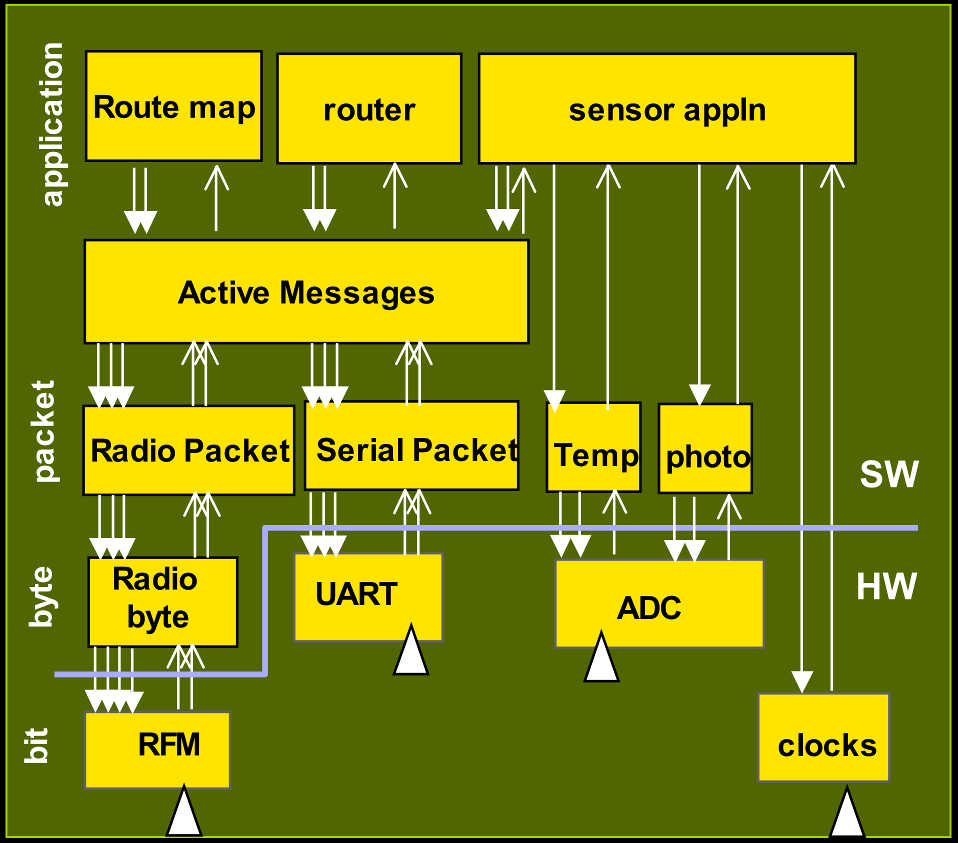


Sensor Network: A set of distributed compute functions cooperating to achieve a set of common goals through interactions with the environment

through a set of distributed sensors
and actuators

TinyOS



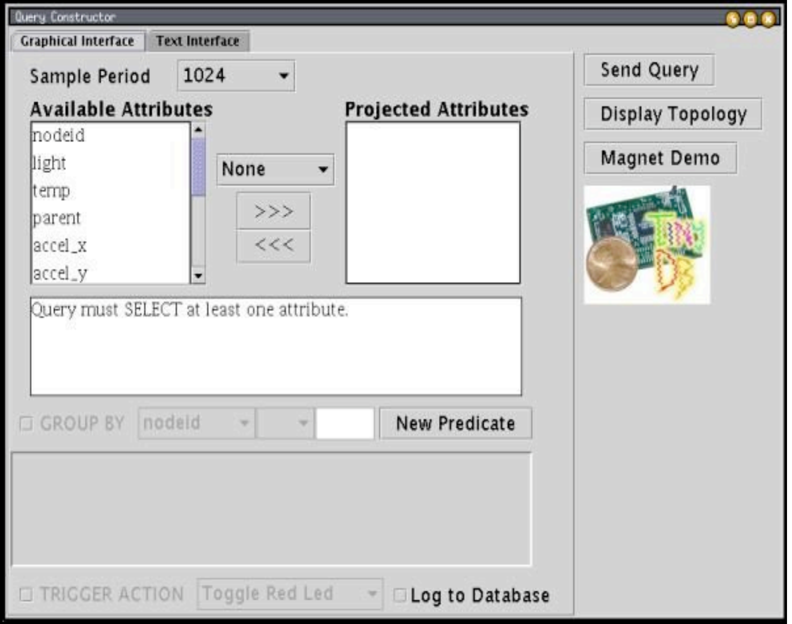


- OS optimized for Sensor Networks
- Used in most existing platforms
- Captures specification as a network of *components* that communicate through an interface of *events* (async) and *commands* (sync)
- Specification using nesC language
- Difficult to use for application and platform developers

TinyDB

- High-level interface (no C programming)
- Defines query service interface *and* its implementation
- Data-centric approach: Sensor Networks queried as Databases
- Declarative SQL-like Queries
- Java-based GUI
- Query example

SELECT temp
FROM sensors
WHERE temp > thresh
TRIGGER ACTION SndPkt
EPOCH DURATION 5 s



Query Constructor

Graphical Interface | Text Interface

Sample Period: 1024

Available Attributes: nodeid, light, temp, parent, accel_x, accel_y

Projected Attributes:

None

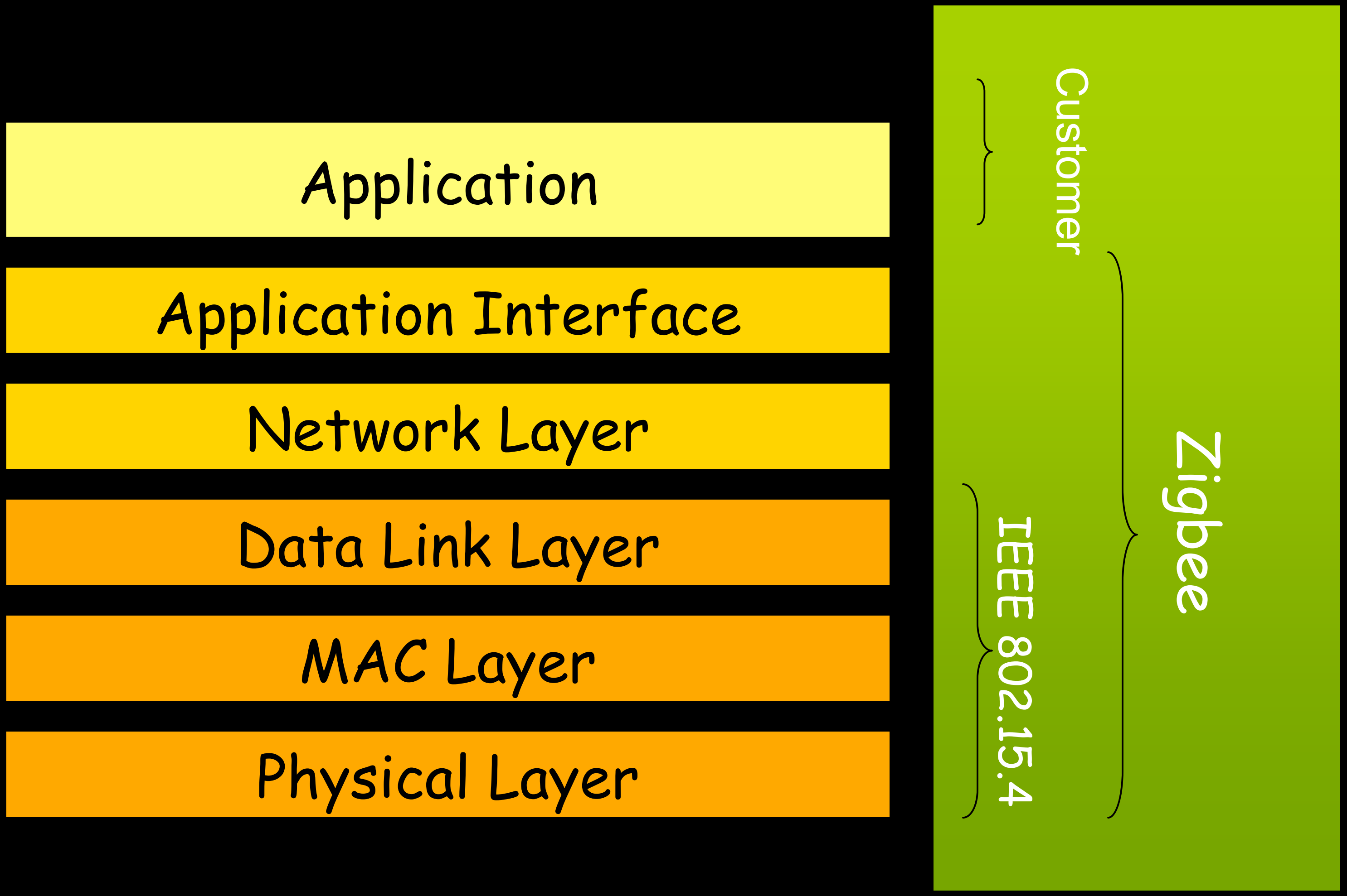
Query must SELECT at least one attribute.

GROUP BY: nodeid

TRIGGER ACTION: Toggle Red Led

Log to Database

Zigbee Alliance

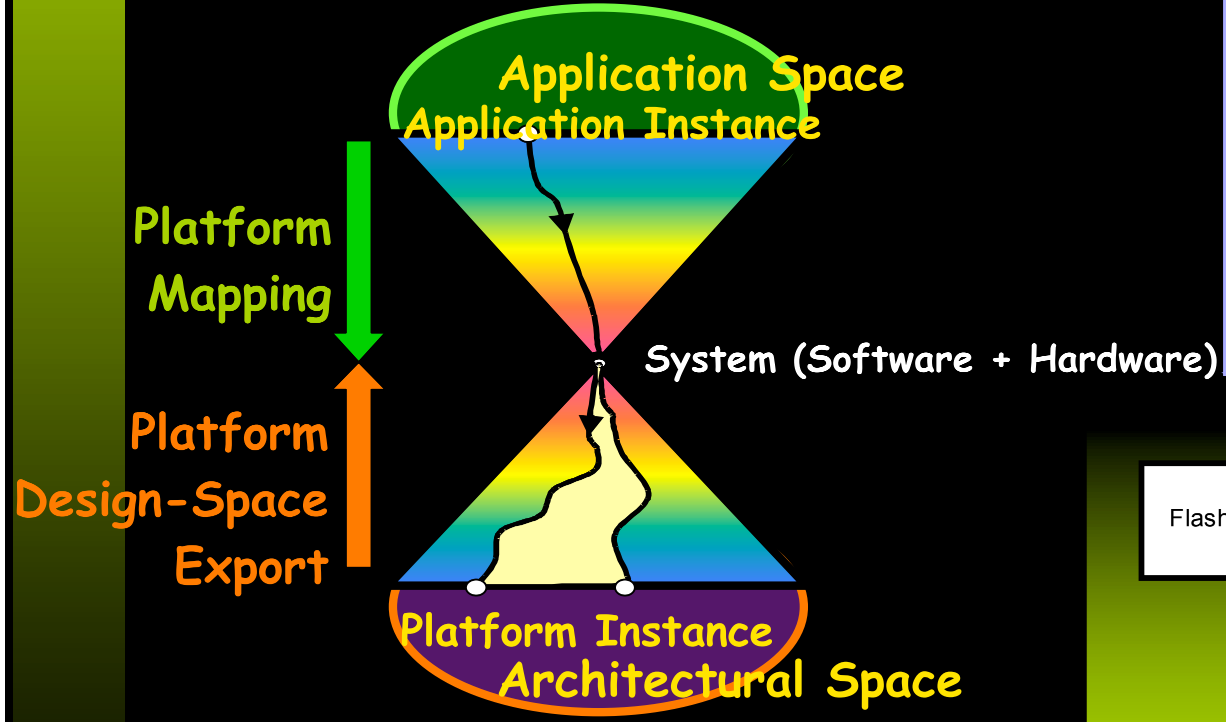


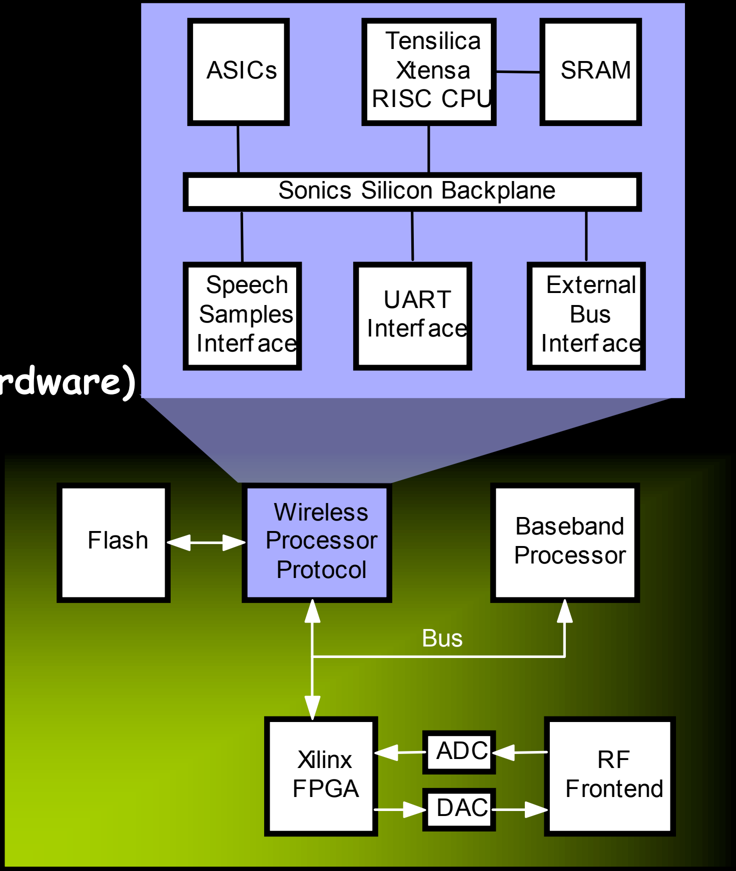
from www.zigbee.org

Outline

- Wireless Sensor Networks and their Evolution
- Platform-based Design for AWSN
  - Sensor Network Service Platform
  - Sensor Network Implementation Platform
- Design Flow
  - Rialto: specification capture (SNPS)
  - Genesis: protocol synthesis (SNIP)

Platform-based Design
(ASV Triangles 1998)





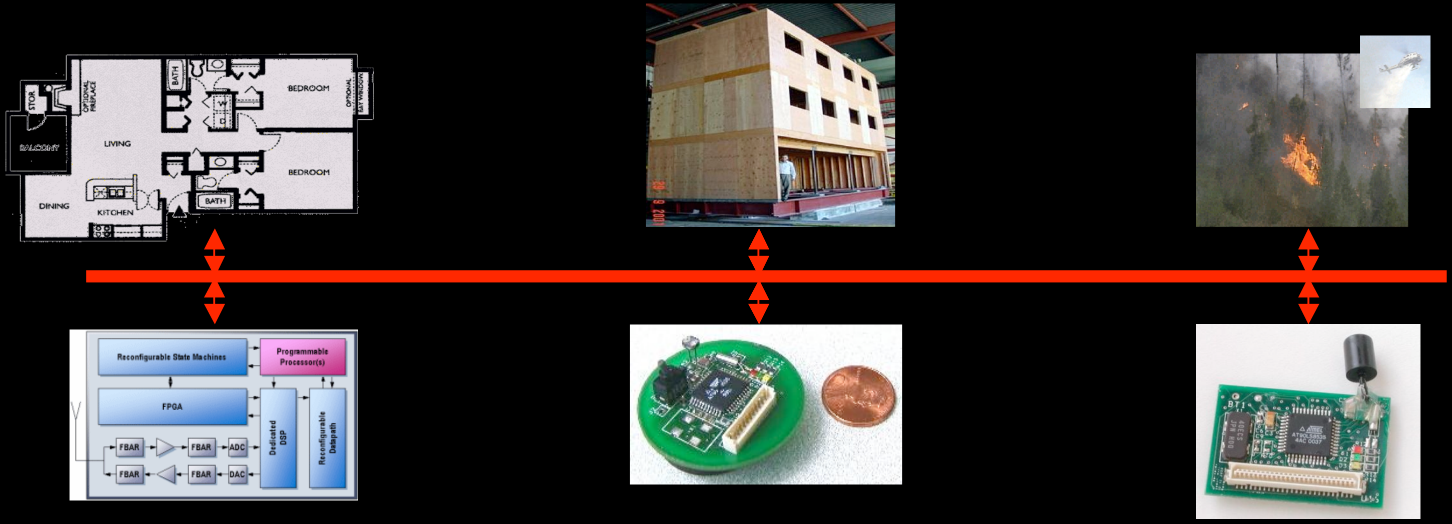
Intercom Platform (BWRC, 2001)

- Platform: library of resources defining an abstraction layer
  - hide unnecessary details
  - expose only relevant parameters for the next step

Principles of Platform methodology: Meet-in-the-Middle

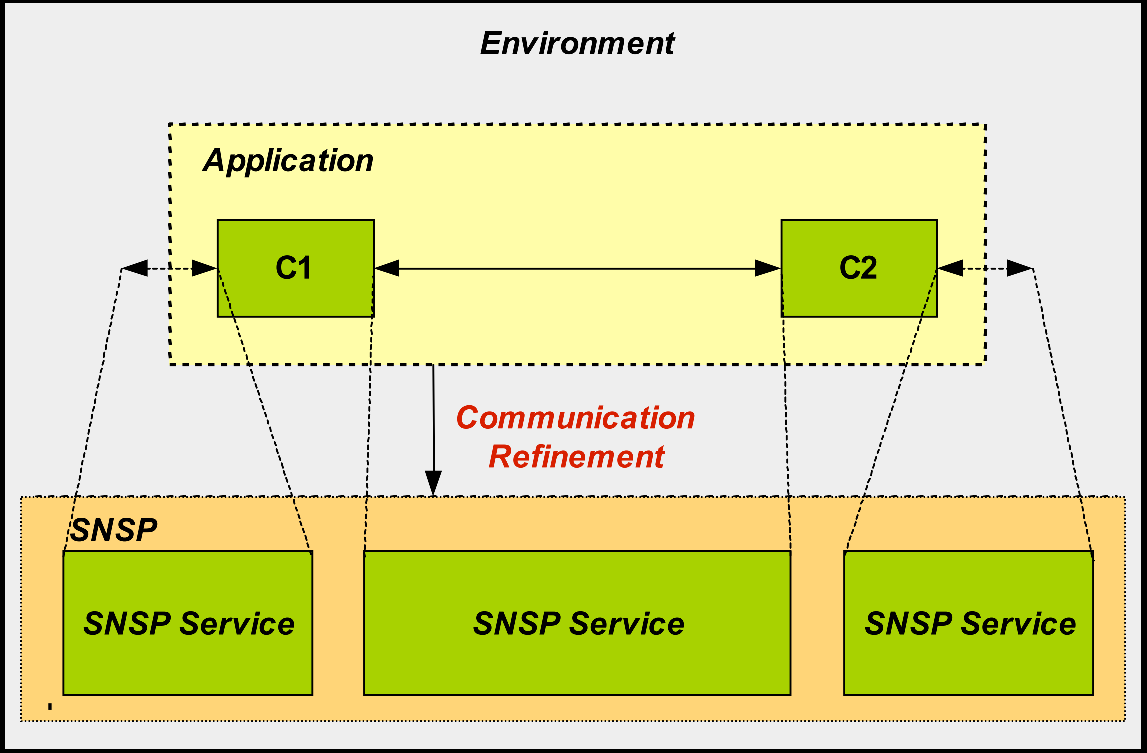
- Top-Down:
  - Define a set of abstraction layers
  - From specifications at a given level, select a solution (controls, components) in terms of components (Platforms) of the following layer and propagate constraints
- Bottom-Up:
  - Platform components (e.g., micro-controller, RTOS, communication primitives) at a given level are abstracted to a higher level by their functionality and a set of parameters that help guiding the solution selection process.
  - The selection process is equivalent to a covering problem if a common semantic domain is used.

A Service-oriented Application Interface



- **Application-level universally agreed Interface**
  - In Internet Sockets support several applications and can be implemented by several protocols
- **Define a standard set of services and interface primitives for Sensor Networks**
  - accessible by the Application (hence called Application Interface)
  - independent on the implementation on any present and future sensor network platform

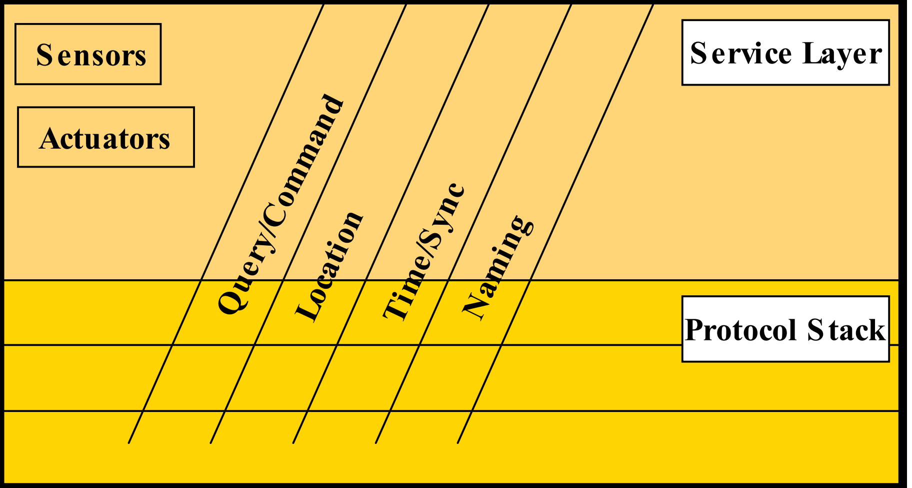
SN Services Platform (SNSP)



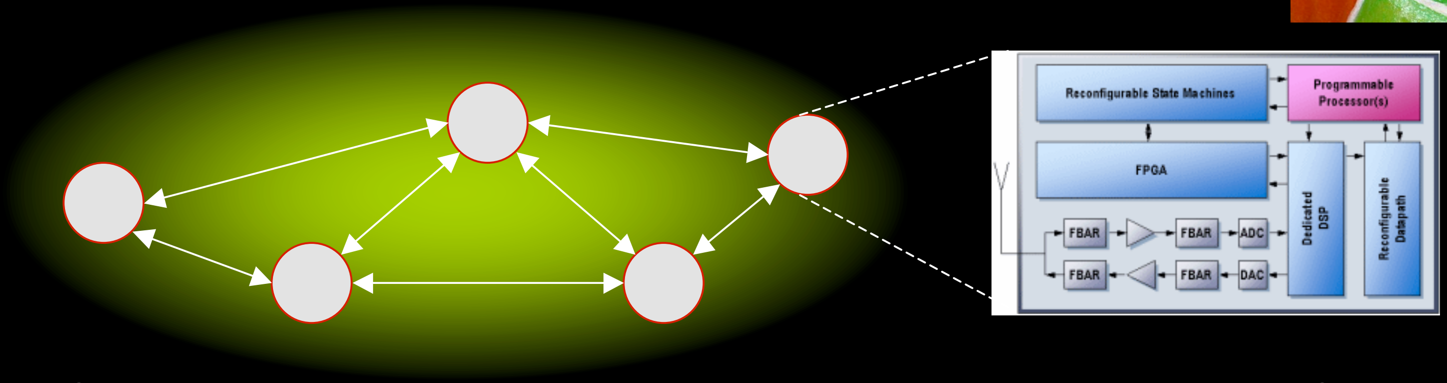
Refines the interaction among controllers and between the controllers and the Environment into interactions between Control, Sensor and Actuation functions

SN Services Platform (SNSP)

- SNSP components:
  - algorithms (e.g. location and synchronization)
  - data processing functions (e.g. aggregation)
  - I/O functions (sensing, actuating)
- Offered Services:
  - Query
  - Command
  - Timing/Synchronization
  - Location
  - Concept Repository
  - Resource Management

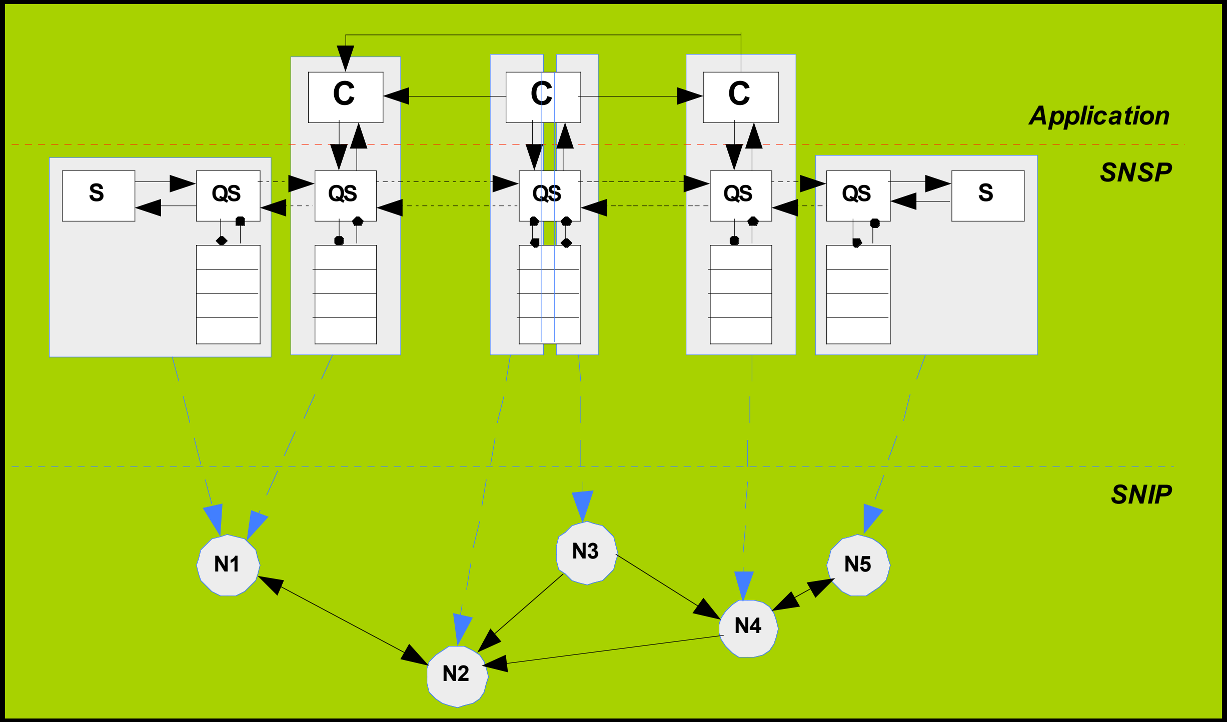


SN Implementation Platform (SNIP)



- Network of interconnected physical nodes that implement the logical functions of the Application and the SNSP
- *Communication protocols* (Routing , MAC)
- *Physical node:* collection of physical resources such as
  - Clocks and energy sources
  - Processing units, memory, and communication and I/O devices
  - Sensor and actuator devices
- *Parameters* of physical nodes:
  - list of sensors and actuators attached to node, memory available for the application, clock frequency range, clock accuracy and stability, level of available energy, cost of computation (energy), cost of communication (energy), transmission rate (range)

Mapping Application and SNSP onto a SNI



- An ***instantiated node*** binds a set of logical Application or SNSP functions to a physical node
- SNIP parameters determine the ***capabilities*** of the network (i.e. quality and cost of the services it provides)

Outline

- Wireless Sensor Networks and their Evolution
- Platform-based Design for AWSN
  - Sensor Network Service Platform
  - Sensor Network Implementation Platform
- Design Flow
  - Rialto: specification capture (SNSP)
  - Genesis: protocol synthesis (SNIP)

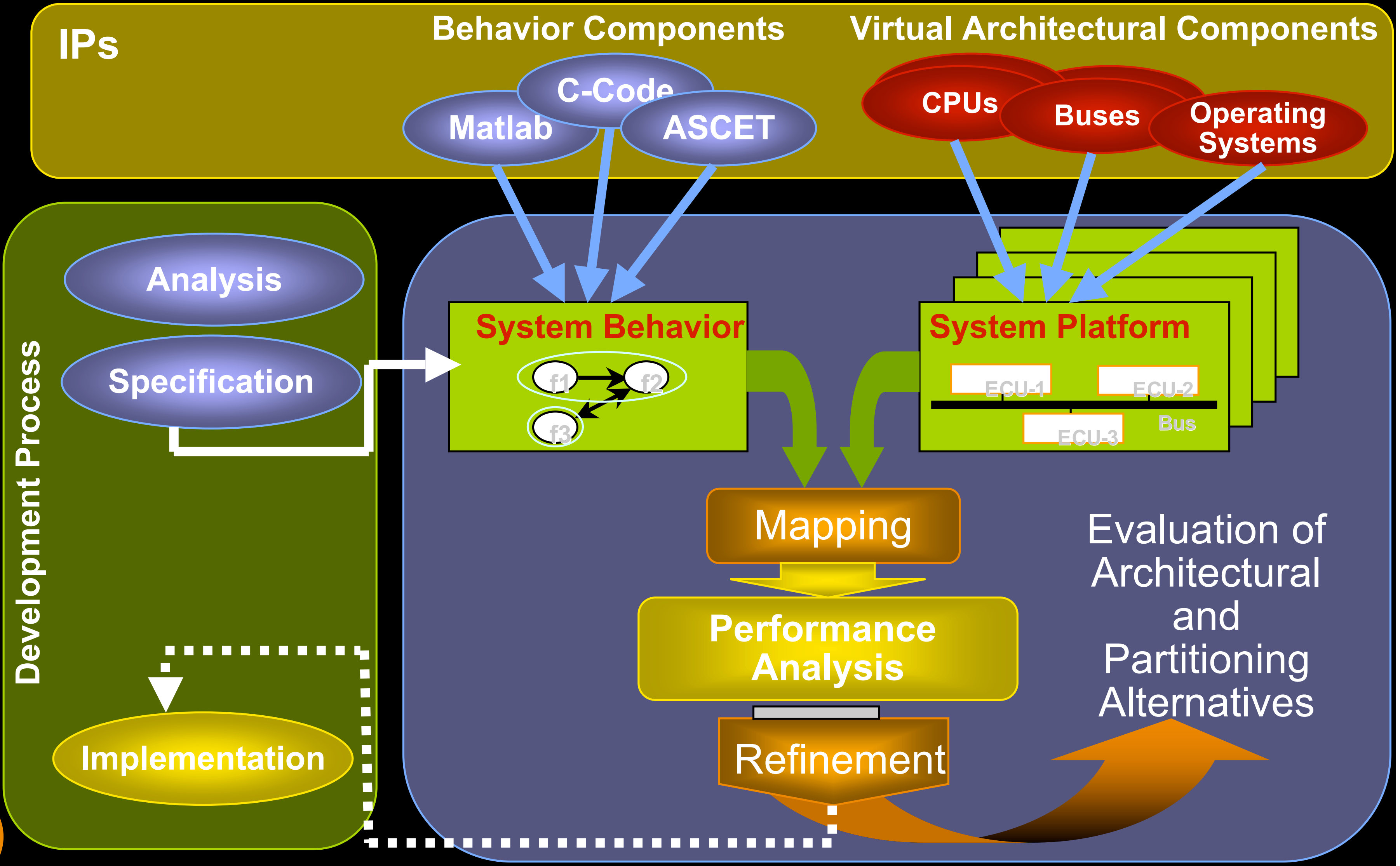
Design Methodology

Short history
- Mostly application driven
- Poor system level perspective
- No established methodology
- Heterogeneity
  - Communication, DSP, Digital, Analog

FORMAL APPROACH
Design Methodology
Design Tools

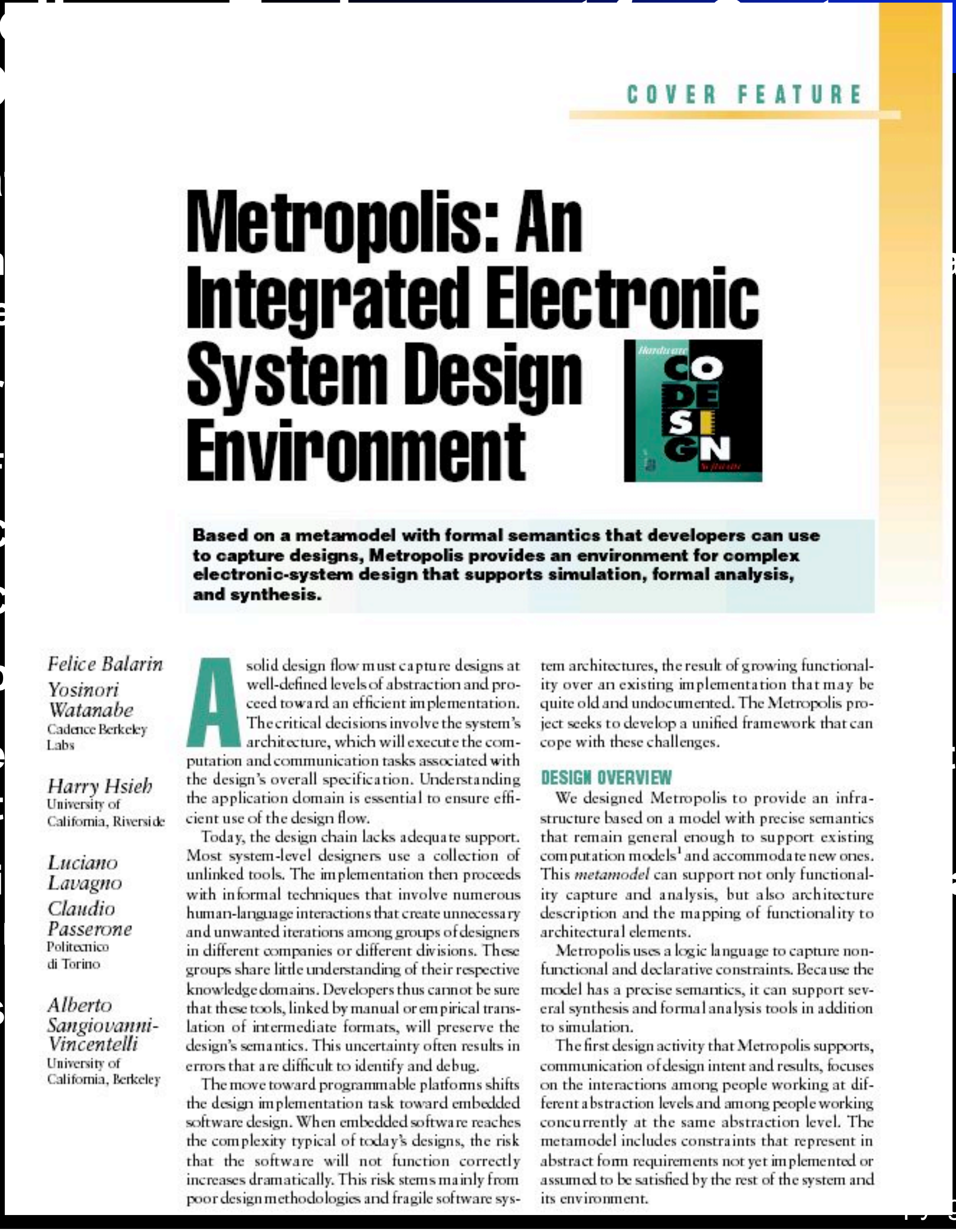
From Lab to Market
- Merging different ingredients
- Implementation Platforms are available
  - Mica, Dust, PicoRadio, Telos
- System Level Design

Separation of Concerns



Metrop
Level D

- Motiva
  - Sem ation is nece
- Platfor
    - F
    - C
    - C
- Metrop
  - Exte tion, and synt
  - Easi ace to exte
- Releas



COVER FEATURE

Metropolis: An Integrated Electronic System Design Environment

Based on a metamodel with formal semantics that developers can use to capture designs, Metropolis provides an environment for complex electronic-system design that supports simulation, formal analysis, and synthesis.

Felice Balarin
Yosinori Watanabe
Cadence Berkeley Labs

Harry Hsieh
University of California, Riverside

Luciano Lavagno
Claudio Passerone
Politecnico di Torino

Alberto Sangiovanni-Vincentelli
University of California, Berkeley

A solid design flow must capture designs at well-defined levels of abstraction and proceed toward an efficient implementation. The critical decisions involve the system's architecture, which will execute the computation and communication tasks associated with the design's overall specification. Understanding the application domain is essential to ensure efficient use of the design flow.

Today, the design chain lacks adequate support. Most system-level designers use a collection of unlinked tools. The implementation then proceeds with informal techniques that involve numerous human-language interactions that create unnecessary and unwanted iterations among groups of designers in different companies or different divisions. These groups share little understanding of their respective knowledge domains. Developers thus cannot be sure that these tools, linked by manual or empirical translation of intermediate formats, will preserve the design's semantics. This uncertainty often results in errors that are difficult to identify and debug.

The move toward programmable platforms shifts the design implementation task toward embedded software design. When embedded software reaches the complexity typical of today's designs, the risk that the software will not function correctly increases dramatically. This risk stems mainly from poor design methodologies and fragile software system architectures, the result of growing functionality over an existing implementation that may be quite old and undocumented. The Metropolis project seeks to develop a unified framework that can cope with these challenges.

DESIGN OVERVIEW

We designed Metropolis to provide an infrastructure based on a model with precise semantics that remain general enough to support existing computation models[1] and accommodate new ones. This *metamodel* can support not only functionality capture and analysis, but also architecture description and the mapping of functionality to architectural elements.

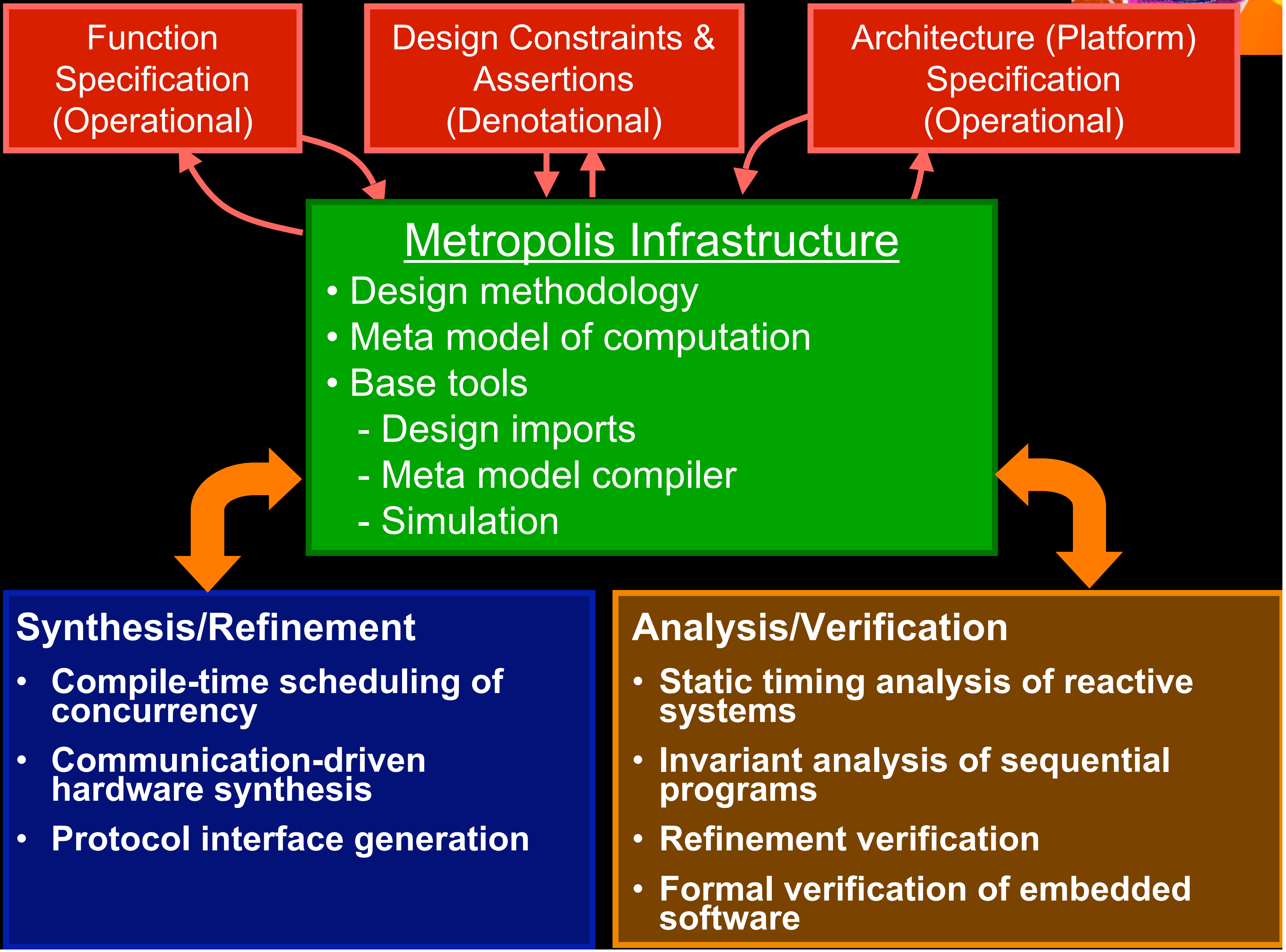
Metropolis uses a logic language to capture nonfunctional and declarative constraints. Because the model has a precise semantics, it can support several synthesis and formal analysis tools in addition to simulation.

The first design activity that Metropolis supports, communication of design intent and results, focuses on the interactions among people working at different abstraction levels and among people working concurrently at the same abstraction level. The metamodel includes constraints that represent in abstract form requirements not yet implemented or assumed to be satisfied by the rest of the system and its environment.

Fundamental Concepts

- Support for different Models of Computation
- Support for Architecture Specification and Analysis
- Mix of imperative and declarative specification styles
- Quantities of interest dictated by the designer, not the framework
- Framework designed to allow interfacing with external tools

Metropolis Framework (Berkeley)



Function Specification (Operational)

Design Constraints & Assertions (Denotational)

Architecture (Platform) Specification (Operational)

Metropolis Infrastructure
- Design methodology
- Meta model of computation
- Base tools
  - Design imports
  - Meta model compiler
  - Simulation

Synthesis/Refinement
- Compile-time scheduling of concurrency
- Communication-driven hardware synthesis
- Protocol interface generation

Analysis/Verification
- Static timing analysis of reactive systems
- Invariant analysis of sequential programs
- Refinement verification
- Formal verification of embedded software

Metropolis Contributors

Industry

- Cadence Berkeley Labs (Design Methods and Tool Development)
- Intel (Communication subsystem of the Centrino platform, Imaging-Video subsystems, Mixed Analog-RF-Digital)
- GM (distributed architecture)
- Infineon (Software defined radios)
- Nokia (Platform-based cell phone design)
- Cypress (Network processor)
- ST (Automotive design)
- United Technology (Security, Conditioning and Elevator Systems)
- Xilinx (Virtex II front-end)

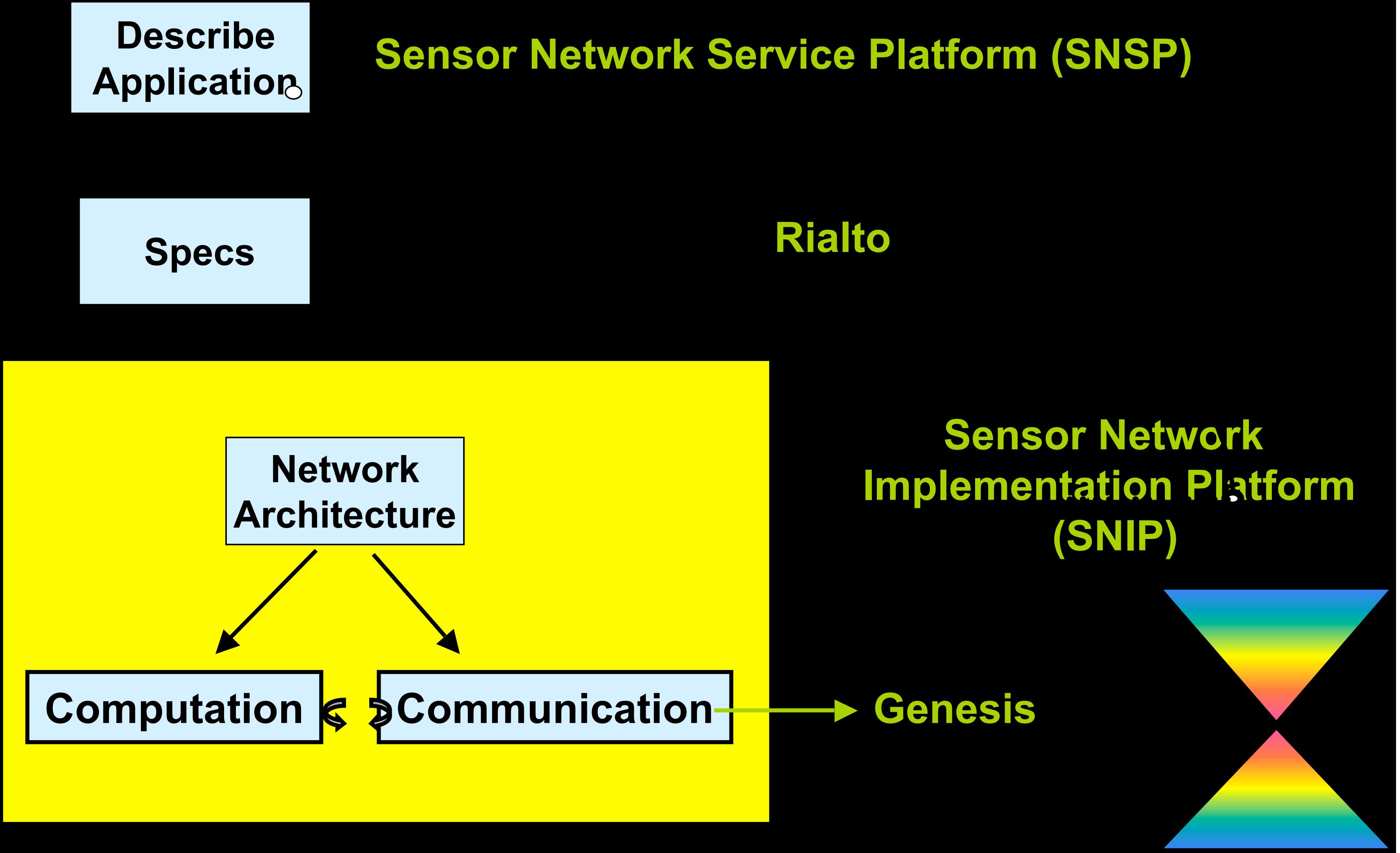
Universities

- University of California at Berkeley
- Carnegie Mellon
- MIT
- UCLA
- Politecnico di Torino
- Universita’ di Pisa
- UC Riverside
- Universitat Politecnica de Catalunya

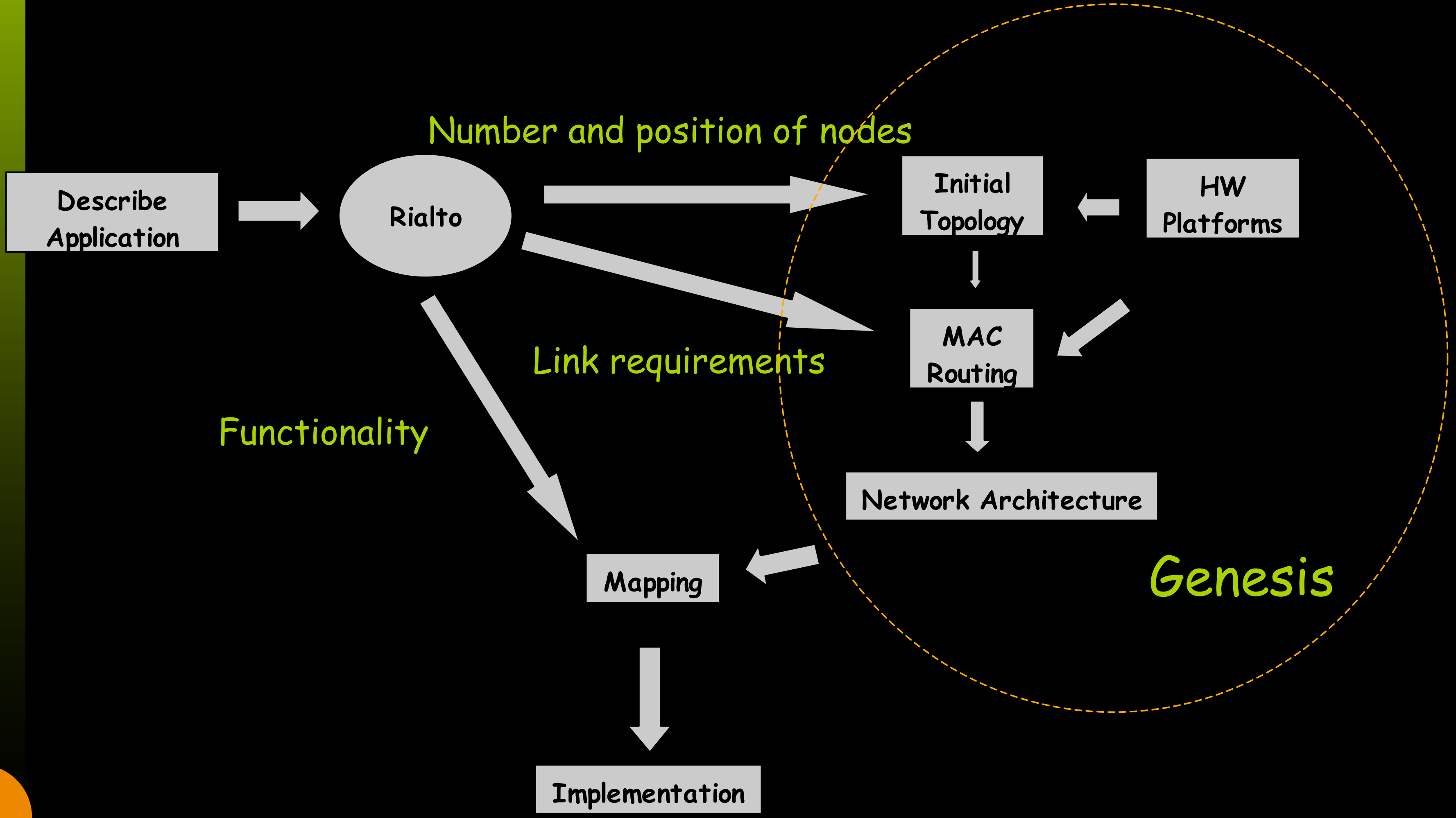
Others

- BWRC
- PARADES

PBD Design Flow


Describe Application
Sensor Network Service Platform (SNSP)
Specs
Rialto
Network Architecture
Sensor Network Implementation Platform (SNIP)
Computation
Communication
Genesis

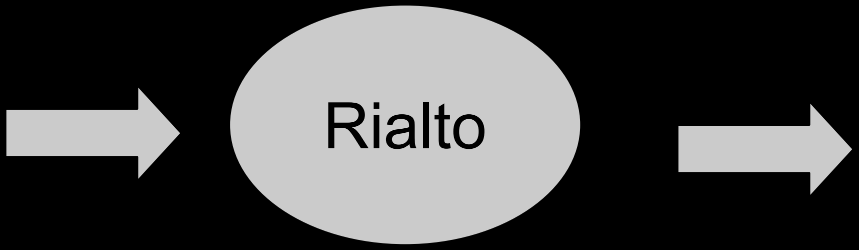
Design Flow


Number and position of nodes
Describe Application
Rialto
Initial Topology
HW Platforms
MAC Routing
Link requirements
Functionality
Network Architecture
Mapping
Genesis
Implementation

Rialto

Allow user to describe the network in terms of logical components queries and services (as in SNSP)

Capture these specifications and produce a set of constraints on LATENCY, ERROR RATES, SENSING, COMPUTATION



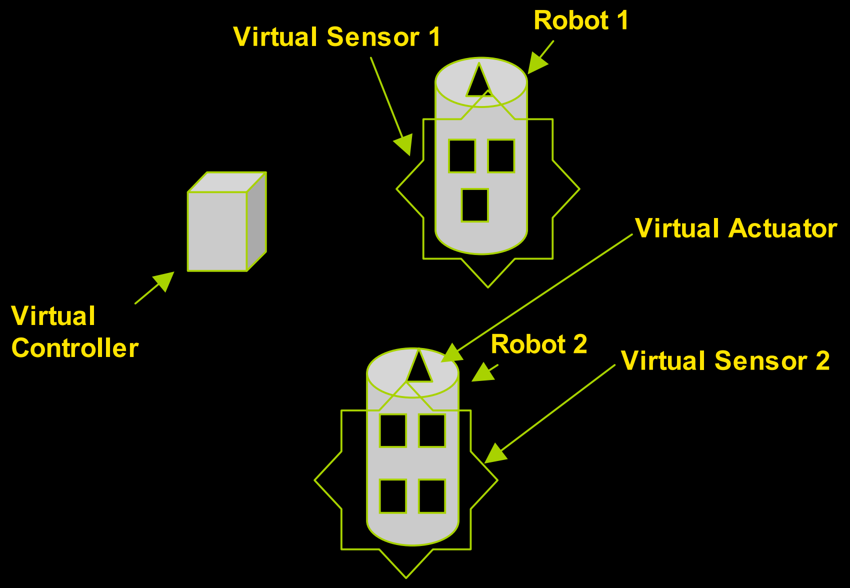
Application Domain

Network Design Domain

Bridging Application with Implementation

Rialto Model

- Three types of “Logical Components”:
  - Virtual Controller
    - Cyclic Control Routine:
      - Queries and Commands
      - Read and Write Semantic
      - Decision Algorithm
  - Virtual Sensor
    - Sensing Capability
      - Read and Write Semantic
  - Virtual Actuator
    - Actuating Capability
      - Read and Write Semantic
- Connections
  - From VC to VS, From VC to VA
  - Unbounded, Bidirectional, Lossless
- Tokens
  - Queries
  - Commands
  - i.e.: “Give me vibration data (average) sampled at a rate R, from time T1 to time T2. Return data within L seconds with a message error rate P”



T=(1,0,Avg,Vib,R,T1,T2,L,P)

Properties of Rialto

- Captures all possible scenario
  - Consider all possible combination of queries and commands
  - Report most “demanding” scenarios
  - Set requirements to satisfy those scenarios
    - Sensing
    - Latency
    - Message Error Rate
- Formal MoC
  - Separation of conditional branches of the controlling algorithm
  - Captures requirements deterministically

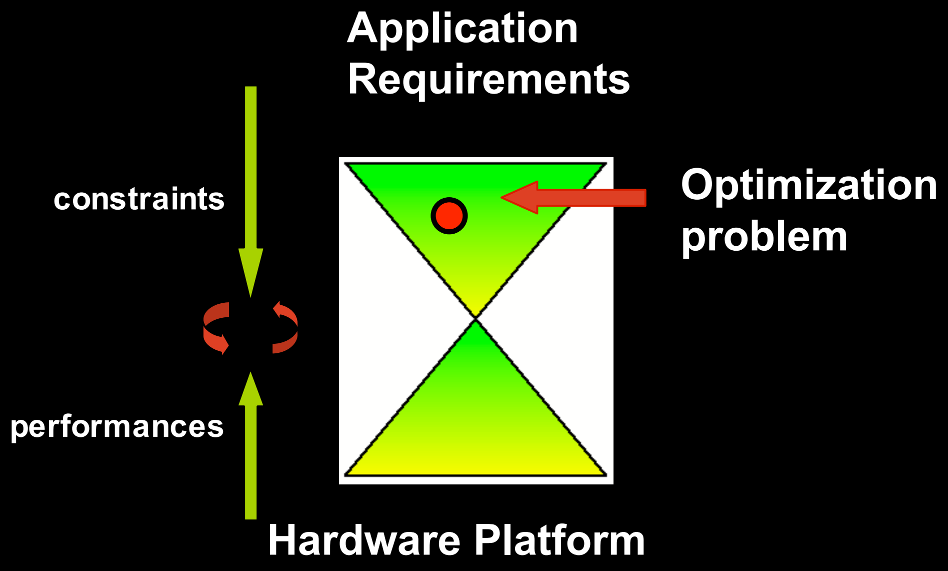
Outline

- Wireless Sensor Networks and their Evolution
- Platform-based Design for AWSN
  - Sensor Network Service Platform
  - Sensor Network Implementation Platform
- Design Flow
  - Rialto: specification capture (SNPS)
  - Genesis: protocol synthesis (SNIP)

Sensor Network Implementation Platform (SNIP)

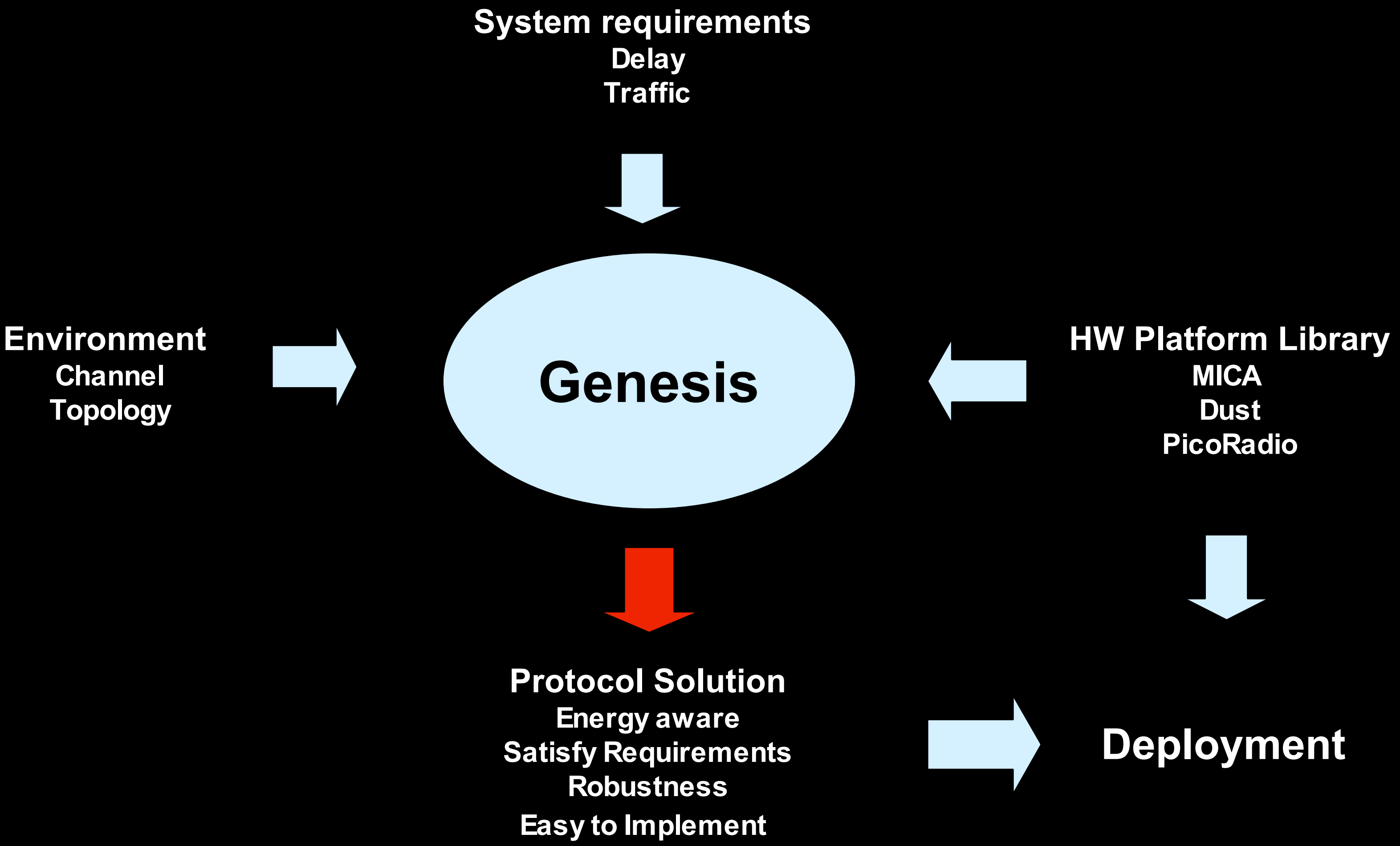
- Sensing, Computation, Communication
  - Satisfy constraints
  - Optimize for energy consumption
  - Orthogonalization of concerns
  - Iterative refinement
    - Start from a valid solution
      - Centralized computation
      - Optimized communication
    - Decentralize Computation
    - Optimize Communication
  - Library
    - Distributed Computation Algorithms
    - Communication Protocols

Protocol Synthesis

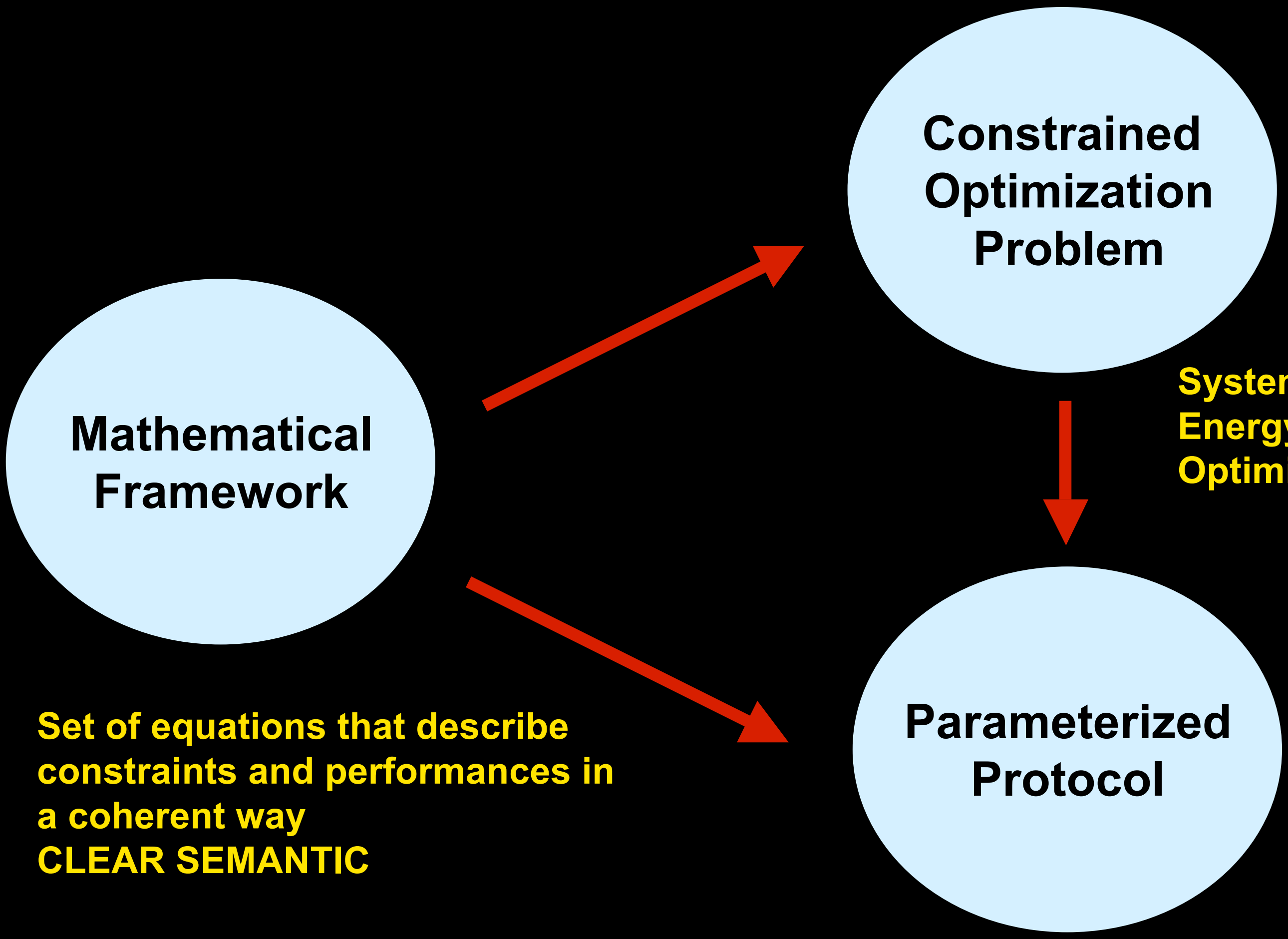


Example ➡ Constraint: End-to-End delay
Cost Function: Energy Consumption
Optimization Space: MAC + Routing

Genesis: Synthesis Engine for Embedded Networks Protocols



Genesis

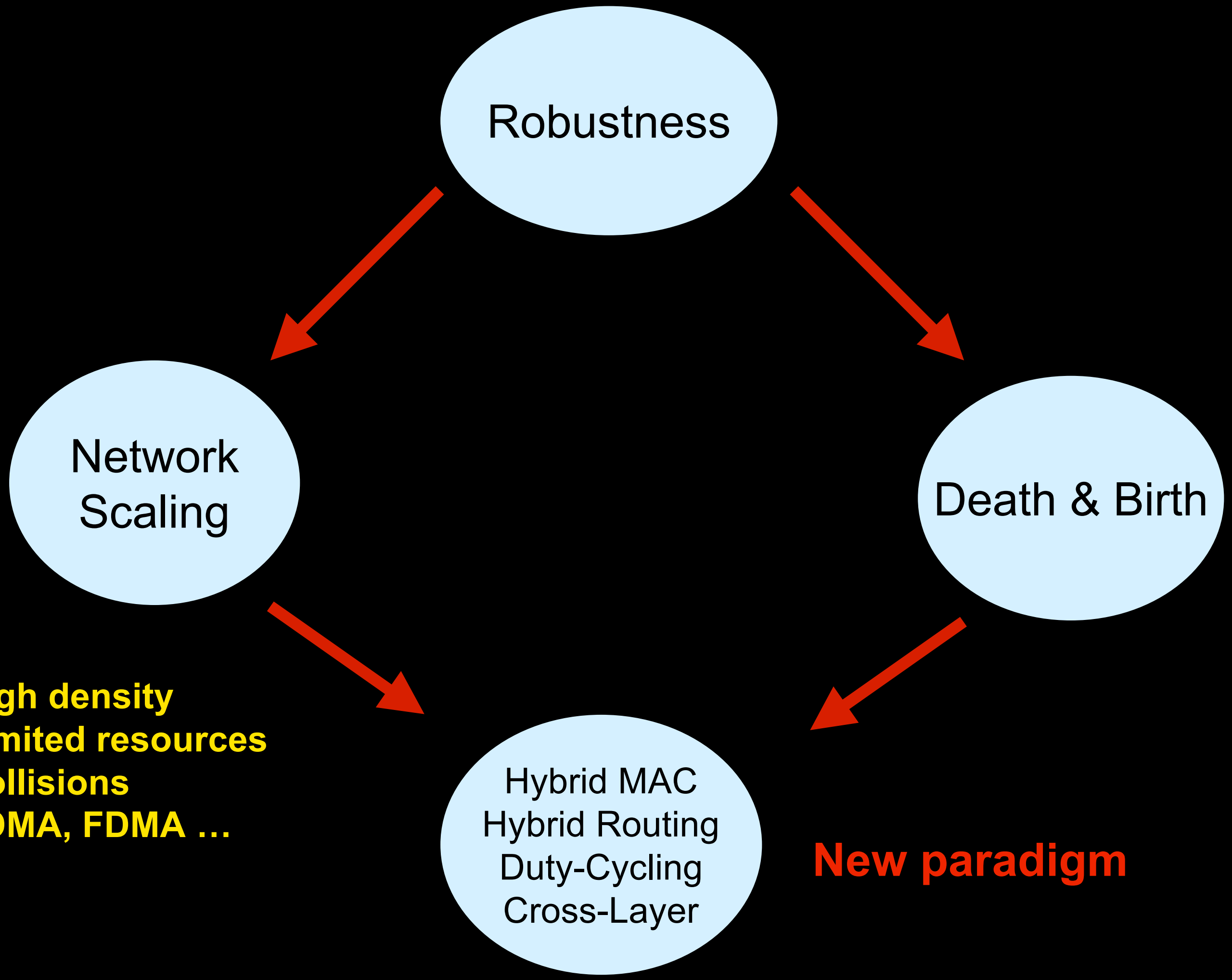


System constraints
Energy is the cost function
Optimize over free parameters

Set of equations that describe constraints and performances in a coherent way
CLEAR SEMANTIC

Simple to implement
Size does matter !!!

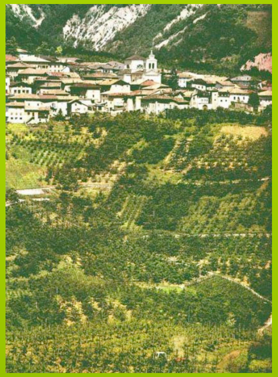
Parameterized Protocol

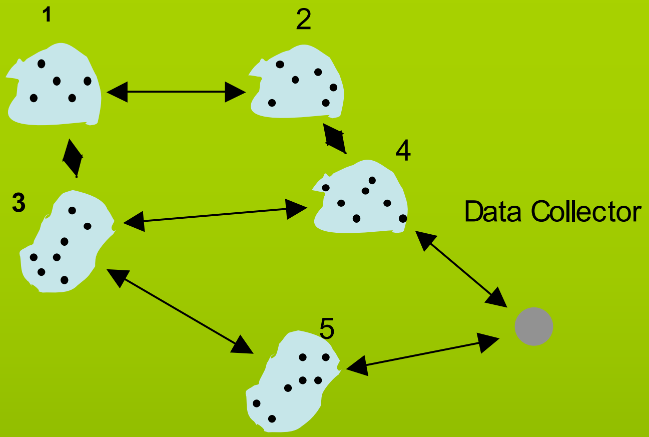


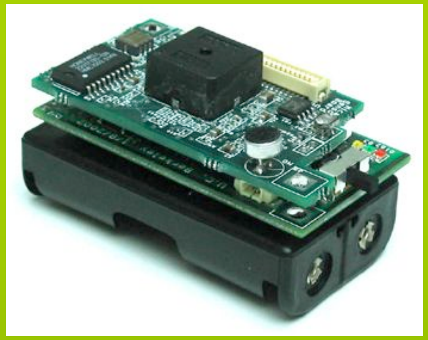
High density
Limited resources
Collisions
TDMA, FDMA …

New paradigm

Example: SERAN



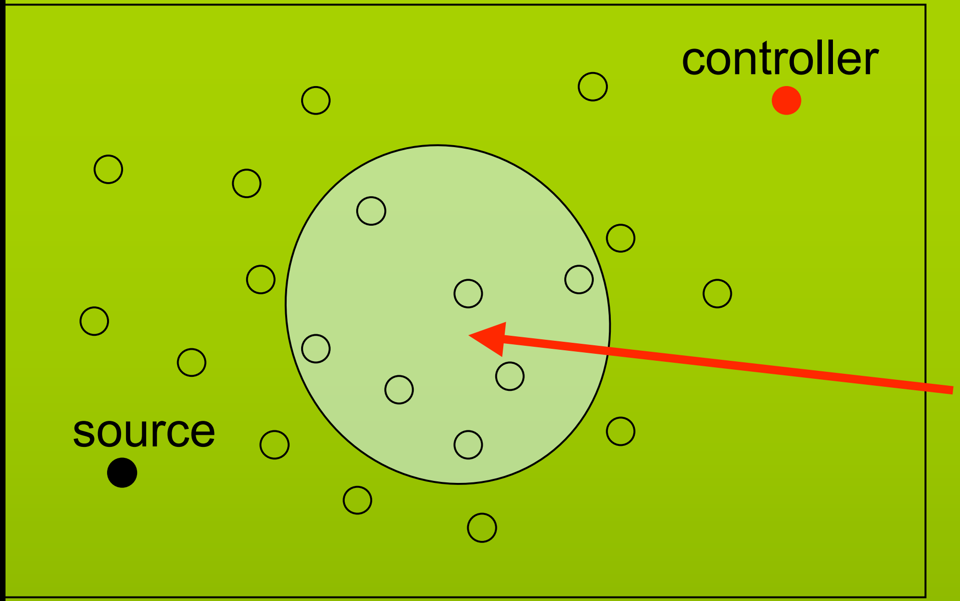




Given:
Topology
Traffic generation requirement
Delay Requirement
Target HW Platform

Generated:
Hybrid Routing
Hybrid MAC
Duty-Cycle
Cross-optimization

Example: Randomized PicoRadio



Density is the main resource:

EXPLOIT EQUIVALENCE

For routing purposes these nodes are equivalent

Given:
Topology
Traffic generation requirement
Delay Requirement
Loss Rate

Generated:
Opportunistic Routing
Randomized MAC
Randomized Duty-Cycle
Cross-optimization
Distributed Adaptation

Applications

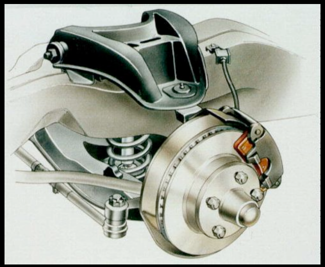
Ambient Intelligence

- CEC

Environmental Monitoring

- Irrigation
- Water pollution

Advanced Sensing



Conclusions

- Wireless Sensor Networks and their Evolution
- Platform-based Design for AWSN
  - Sensor Network Service Platform
  - Sensor Network Implementation Platform
- Design Flow
  - Rialto: specification capture (SNPS)
  - Genesis: protocol synthesis (SNIP)