Imagine a *networked* world of...

... sensors, actuators, processors and storage, which is part of a shared physical infrastructure.

Not hard to imagine!

---

### Sensorium Infrastructure @ BU

**Sensorium:**

A common space equipped with video sensors (VS) for ubiquitous recognition and tracking of activities therein.

**Infrastructure:**

- Range of VS Elements
- Programmable VS Network
- Backend compute engines
- Backend TByte storage
- Mobile/wireless query units
- Research Engineer

---

### Sensoria: A Paradigm Shift

- The proliferation of networked, embedded, and mobile digital video sensors requires a paradigm shift in many areas of CS to address:
  - The unique spatio-temporal aspects of sensory (video) data acquisition, processing, representation, communication, storage, real-time indexing/retrieval, data mining
  - The challenges of programming, QoS management and coordinated resource arbitration of sensory networks, which are both embedded and mobile

**The other extreme in sensor networks research!**

---

### Sensoria: Example Deployments

- **Assistive Environments**
  - *e.g.* for home/hospice/elder care/…

- **Safety Monitoring**
  - *e.g.* in factories/daycare/hospitals/garages/subway/…

- **Intelligent Spaces**
  - *e.g.* for classrooms/meeting rooms/theaters/farms/…

- **Secure Facilities and Homeland Security Uses**
  - *e.g.* at airports/embassies/prisons/…

- **People Flow/Activity Studies**
  - *e.g.* at retail stores/museums/…

---

### snBench

The Sensorium is the computer...

Design/implement the programming and run-time infrastructure necessary for developers to specify and deploy truly distributed applications over a heterogeneous network of Sensing Elements (SEs) and Computing Elements (CEs).

---
snBench: Goals

- "Write Once, Run... Wherever"
- Don’t program nodes...
  - Program the network!
    - Start with building blocks
    - Sensors
    - Stock alg (edge detect, face count, FFT)
    - Glue together with high-level language
      - Conditionals, loops, functions
    - Pretend the network isn’t there
      - "Single System Image"

snBench: Programming Cycle

- Program
  - Program specified by gluing together building blocks using SNAFU language
- Compile
  - SNAFU program is compiled to produce a plan of execution expressed in STEP
- Map and Link
  - STEP plans are decomposed in smaller dispatch-able STEPs which are linked
- Load and Execute
  - STEP plans are dispatched (i.e., loaded) into SXE execution environments

SNAFU: SNet Apps as FUnctions!

- Functional specification language
  - e.g., identify the face seen through camera 1
    identify(facefind(snapshot(cam1)))

SNAFU: Events as eval triggers

- e.g., motion detected at night triggers email
  - email("security@bu.edu",
    trigger(
      {motion(snapshot(cam2)) && (2am<NOW<4am), snapshot(cam2)})
SNAFU: Level Trigger

- "As long as p is true re-evaluate a"
  
  ```
  level_trigger(p, a)
  while(true)
  if (p) return (a);
  ```

- Level triggers are persistent queries that return a stream of evaluations of "a" while "p" is true

SNAFU: Edge Trigger

- "When p becomes true re-evaluate a"
  
  ```
  edge_trigger(p, a)
  while(true)
  if (p) {return(a); while(p) {}};
  ```

- Edge triggers are persistent queries that return a stream of evaluations of "a" every time "p" transitions from false to true

SNAFU: Annotating triggers

- Level triggers could be made periodic
  
  ```
  period(100ms, level_trigger(motion?(cam1), snapshot(cam1)))
  ```

SNAFU: Accessing trigger results

- Three “read” semantics:
  - Non-blocking read – Last result is returned
  - Blocking read – Wait for next result
  - Fresh read – Wait for a “from scratch” result

STEP: Sensorium Typed Exec Plan

- "Instruction Set Architecture" is a DAG

- Notes:
  - "Evaluation" percolates up from leaf nodes toward roots.
  - Sensor nodes provide fresh data at all times.
  - Function nodes evaluate once then disable themselves.
  - Trigger nodes re-enable children for re-evaluation.
STEP: XML Representation

Serialized STEP tree represented using XML

```
<root>
  <entity id="01"
     xmlns="http://www.igs.org/step"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.igs.org/step">
    <product_definition>
      <name>Product</name>
      <description>Document Description</description>
      <instancecname>Product_Name</instancecname>
    </product_definition>
    <feature_definition>
      <name>Feature_Name</name>
      <description>Feature Description</description>
      <instancecname>Feature_Name</instancecname>
    </feature_definition>
  </entity>
</root>
```

snBench: Runtime Setting

- SNAFU-to-STEP compiler
- SXE: Sensorium eXecution Environment
- SSD: Sensorium Service Dispatcher

Map, Link, and Dispatch

Map & Link Challenges
- How did we arrive at this partitioning?
- How did we arrive at these assignments?

Why snBench?

- A Sensorium Research Catalyst
  - Intrinsic research
    - Languages, type systems, compilers, graph algorithms, schedulers, QoS engines, naming, etc. to make the framework efficient/useful/flexible
  - Extrinsic research
    - Accessible framework for developing distributed sensing applications in heterogeneous networks

Intrinsic Research: Scheduling

- Sensorium Resource Manager
  - Keeps track of CPU loading conditions

Scheduling: Processing Capacity
Intrinsic Research: Scheduling

- Scheduling amounts to a constrained graph embedding problem
  - Difficult (NP-hard) Problem
  - Approximations and heuristics

Different programs may share STEP sub-graphs

- Examples:
  - `email("security@bu.edu", trigger((motion(snapshot(cam2)) && (2am<NOW<4am)), snapshot(cam2)))`
  - `trigger(facerecognizer, facelibrary("Azer Bestavros"), email("azer.wife@cs.bu.edu", "He's working late again."))`

STEP Scheduling: CSE

Common Subexpression/Subgraph Elimination
**STEP Scheduling: CSE**

```
let x = snapshot(cam2) in
email("security@bu.edu", block(edge_trigger(motion(x) && (2am<NOW<4am), x)))
```

**Intrinsic Research: Compilation**

- **Program Optimization**

  - `let a = snapshot(cam2) in
    email("security@bu.edu", (2am<NOW<4am) && true)
    email("azer.wife@cs.bu.edu", "He's working late again.")`

**Intrinsic Research: Virtualization**

- **What if we can’t run an SXE on a node?**
- **STEP: as a virtual ISA**
  - Embedded C dialects?
  - Java ME?
- **Alternate linking protocols**
  - Serial (base station)
  - SN wireless protocols (e.g. 802.15.4)

**Intrinsic Research: Expressive PL**

- **Better Programming Languages**
- **Alternate Execution Environments**

**snBench: Status**

- **Skeleton of snBench is done**
  - Basic SNAFU-STEP compiler
  - Basic STEP dispatcher
  - Uses HTTP to exchange STEPs in XML
  - Manual binding of STEP nodes to resources
  - Basic SXEs implemented and functional for
    - Generic computing elements (Java classes)
    - Video sensor nodes (webcams mounted in laboratories)
    - Berkeley Motes (temperature, etc.)
  - Limited (but growing) library of “opcodes”
    - Grab a frame, frame diff, face count, temperature/time stamp, …

- **snBench Live**
  - Active Sensorium SXEs and STEP graphs (Demo)