*sn*BENCH

A Development and Run-Time Platform for Rapid Deployment of Sensor Network Applications

Azer Bestavros
Assaf Kfoury & Adam Bradley & Michael Ocean

Computer Science Department
Boston University

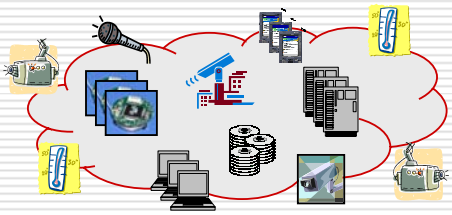
http://www.cs.bu.edu/groups/rwing

The iBench Initiative

Basenets 2005: IEEE International Workshop on Broadband Advanced Sensor Networks
October 3, 2005 / Boston, Massachusetts

Imagine a *networked* world of …

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

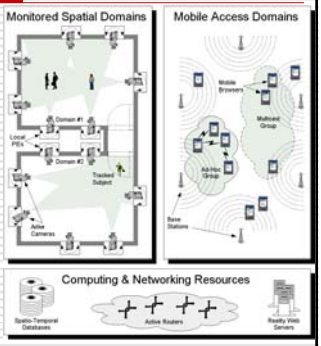


Not hard to imagine!

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…

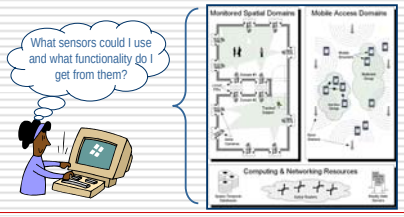
Secure Facilities and Homeland Security Uses
- *e.g.* at airports/embassies/prisons/…

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

*sn*Bench

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 of Computing Elements (CEs)



*sn*Bench: Goals

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

*sn*Bench: 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: SNet Apps as FUnctions!

- Use "annotative" functions for constraints
  - *e.g., what streams can be "exposed"*

```
public(facecount(snapshot(private(cam1))))
```

  - *e.g., minimum acceptable resolution*

```
facecount(resn(800x600x24b,snapshot(cam1)))
```

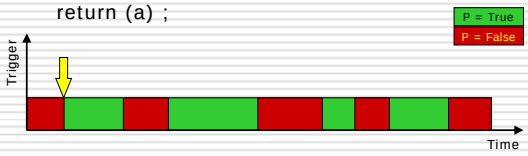

SNAFU: Events as eval triggers

- Events in time captured by "triggers"
  - *e.g., motion detected at night triggers email*

```
email("security@bu.edu",
  trigger(
    (motion(snapshot(cam2)) && (2am<NOW<4am)),
    snapshot(cam2)))
```


SNAFU: "Do once" Trigger

- "Wait until **p** becomes true to evaluate **a**"
  trigger(p,a)
  ➔ do {} until (p) ;
  return (a) ;



- Once fired, evaluation of trigger is complete

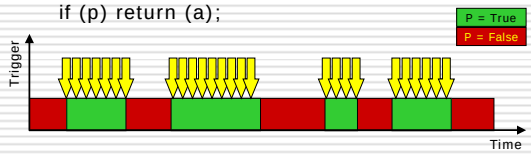
SNAFU: Level Trigger

- "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

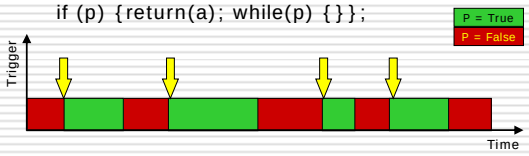
SNAFU: Edge Trigger

- "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

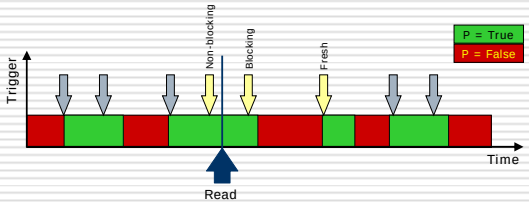
- Level triggers could be made periodic

  This is an example of scheduling annotations

```
period(100ms,
  level_trigger(motion?(cam1),snapshot(cam1))
  )
```


SNAFU: Accessing trigger results

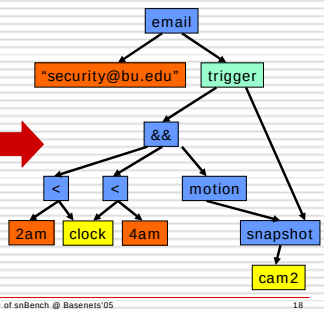
- Three "read" semantics:
  - Non-blocking read - Last result is returned
  - Blocking read - Wait for next result
  - Fresh read - Wait for a "from scratch" result



STEP: Sensorium Typed Exec Plan

"Instruction Set Architecture" is a DAG

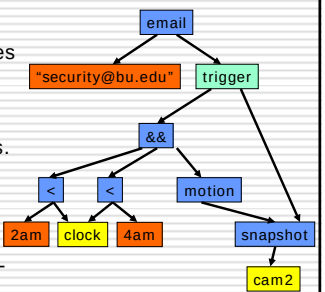
```
email("security@bu.edu",
  trigger(
    (motion(snapshot(cam2))
     && (2am<NOW<4am)),
    snapshot(cam2)))
```



STEP: Sensorium Typed Exec Plan

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

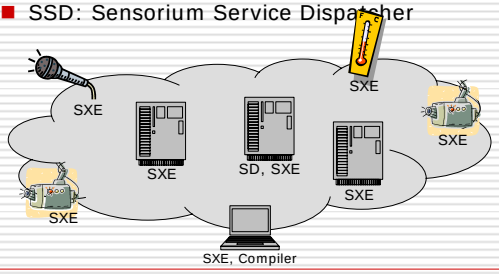
Serialized STEP tree represented using XML

```
max(facecount(snapshot(sensor("s05","cam1"))),
    facecount(snapshot(sensor("s05","cam2"))))
```

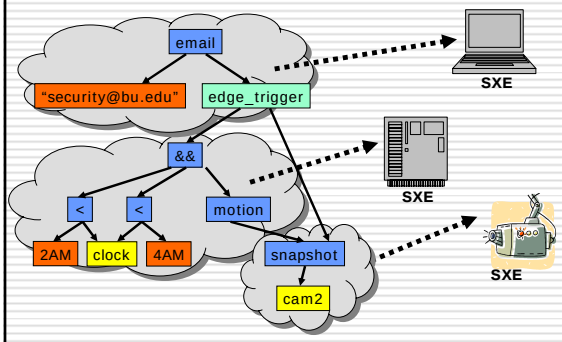
```
<step id="202219@s00.sensorium.bu.edu">
  <exp opcode="max" id="abcd" bindto="http://s05.sensorium.bu.edu:8080">
    <flowtype name="persist" value="Mon Jul 25 23:59:59 EDT 2005" />
    <exp opcode="facecount" id="bcde" bindto="http://s05.sensorium.bu.edu:8080">
      <exp opcode="snapshot" id="cdef"><value id="defg">
        <sensor type="snbench/image">
          http://s05.sensorium.bu.edu:8080/snbench/sxe/sensor/image/1
        </sensor></value></exp></exp>
    <exp opcode="facecount" id="efgh" bindto="http://s05.sensorium.bu.edu:8080">
      <exp opcode="snapshot" id="fghi"><value id="ghij">
        <sensor type="snbench/image">
          http://s05.sensorium.bu.edu:8080/snbench/sxe/sensor/image/2
        </sensor></value></exp></exp>
  </exp>
</step>
```


*sn*Bench: Runtime Setting

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



Map, Link, and Dispatch

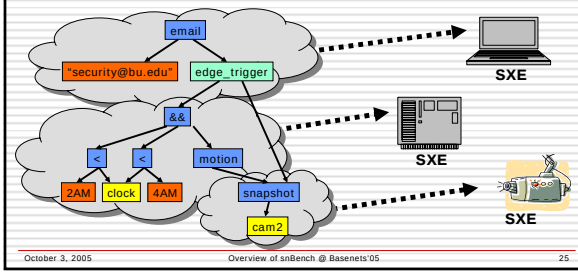


Why *sn*Bench?

- 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

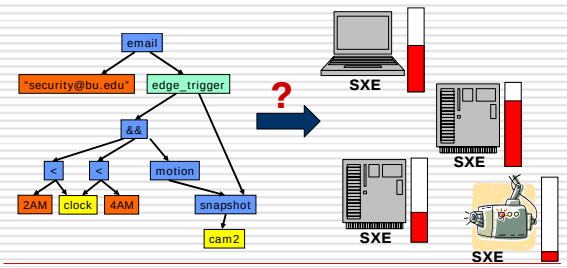
Intrinsic Research: Scheduling

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



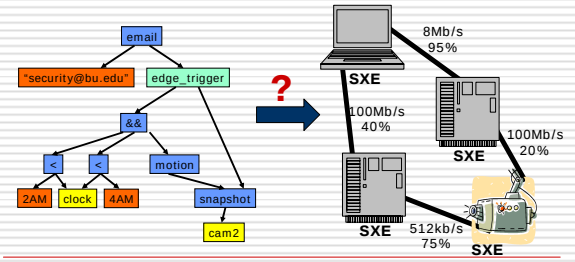
Scheduling: Processing Capacity

- Sensorium Resource Manager
  - Keeps track of CPU loading conditions



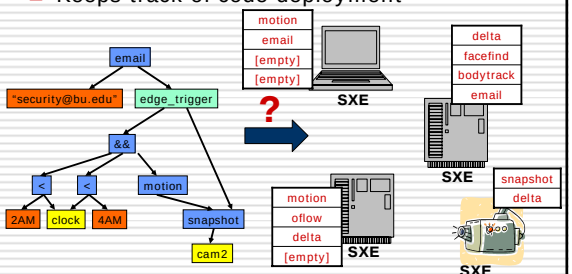
Scheduling: Network Capacity

- Sensorium Resource Manager
  - Keeps track of current network conditions



Scheduling: Node Capabilities

- Sensorium Resource Manager
  - Keeps track of code deployment



Intrinsic Research: Scheduling

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

Intrinsic Research: Scheduling

Different programs may share STEP sub-graphs

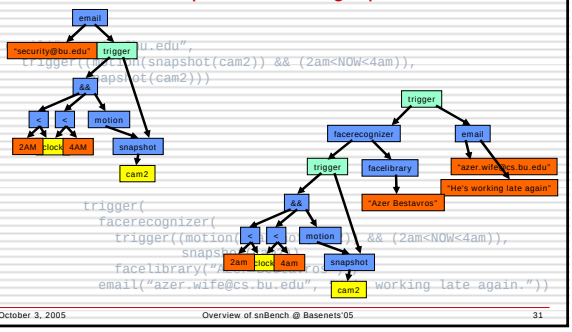
- Examples:

```
email("security@bu.edu",
  trigger((motion(snapshot(cam2)) && (2am<NOW<4am)),
          snapshot(cam2)))
```

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

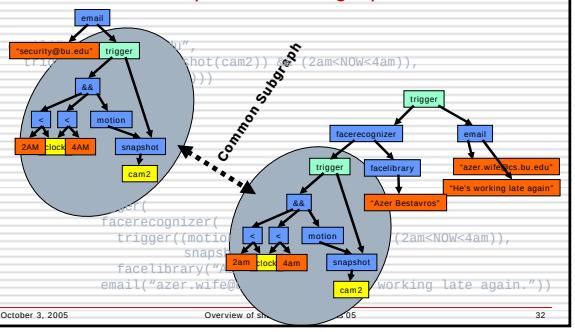

STEP Scheduling: CSE

Common Subexpression/Subgraph Elimination



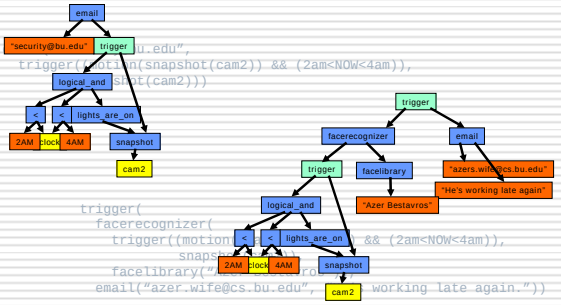
STEP Scheduling: CSE

Common Subexpression/Subgraph Elimination



STEP Scheduling: CSE

Common Subexpression/Subgraph Elimination



Intrinsic Research: Compilation

- Program Optimization

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

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

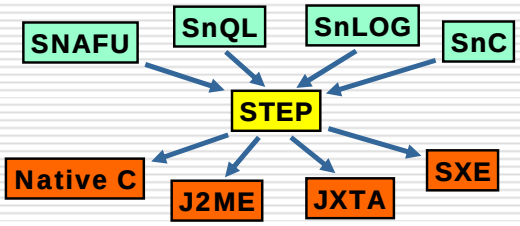
- What to optimize?
  - STEP Node count?
  - Trigger count?

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



*sn*Bench: Status

- Skeleton of *sn*Bench is done
  - Basic SNAFU-STEP compiler
    - No annotative functions support
  - 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, …
- *sn*Bench Live
  - Active Sensorium SXEs and STEP graphs (Demo)

*sn*BENCH

A Development and Run-Time Platform for Rapid Deployment of Sensor Network Applications

Azer Bestavros

Assaf Kfoury & Adam Bradley & Michael Ocean

Computer Science Department

Boston University

http://www.cs.bu.edu/groups/wing

The iBench Initiative

Basenets 2005: IEEE International Workshop on Broadband Advanced Sensor Networks

October 3, 2005 / Boston, Massachusetts