Global-scale Reactive Sensing Systems

Ramesh Govindan

University of Southern California

ramesh@usc.edu

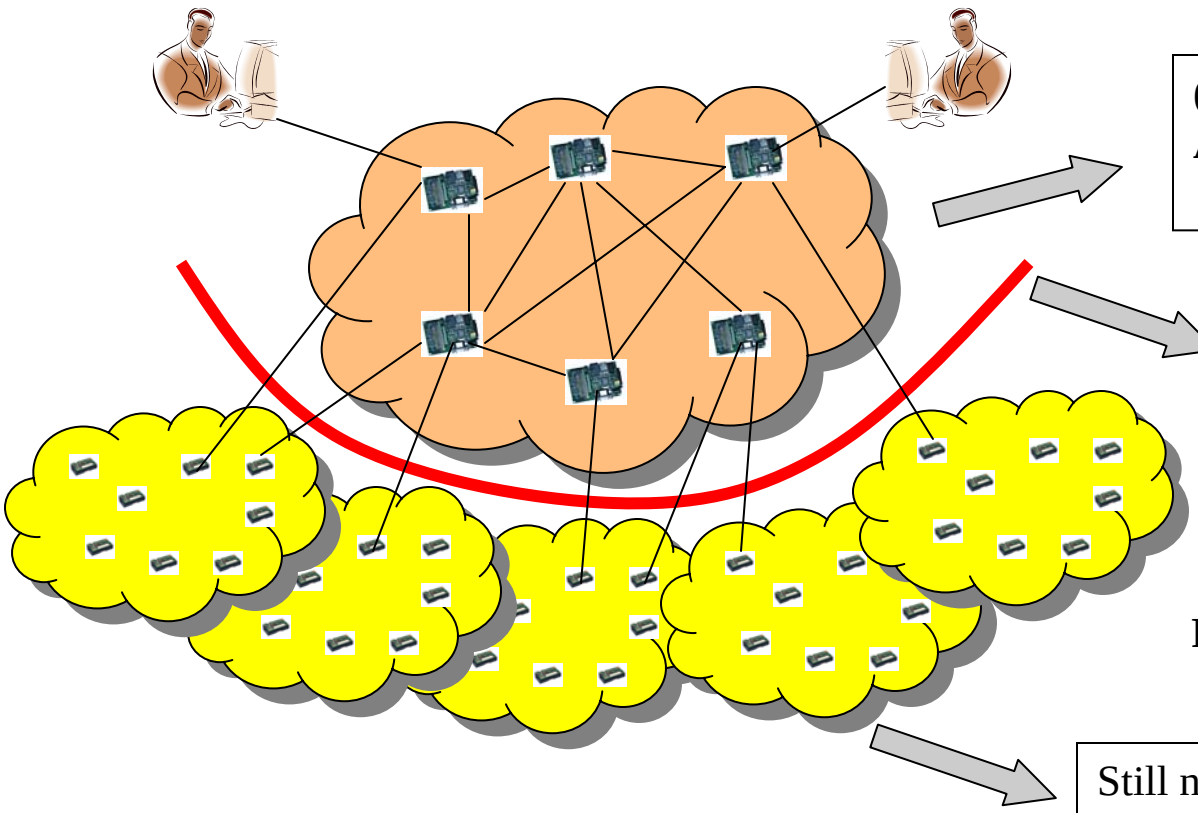
Where we are

```
wisden_tenet.c        Wed Jul 12 09:24:17 2006        1
#include "transportAPI.h"
#include "task_construct.h"

char *task_string = "mda400(20000,0,0x101,0x202,0x303,4,20)->onset_detector(0x303,0x313)-
>sendstr()";

int main(int argc, char **argv)
{
    unsigned char b[200];
    int l, tid = 0;
    uint16_t r_tid, r_addr;

    /* construct a task packet */
    l = construct_task(b, task_string);
    if (l < 0) {
        printf("task description error!! \n");
        exit(1);
    }

    /* send the task */
    tid = send_task((uint16_t)tid, l, b);
    if (tid > 0) {
        printf("\ntask has been sent with tid %d!!\n", tid);
    } else {
        exit(1);
    }

    /* receive response packets */
    while(1) {
        unsigned char *packet = receive_packet(&r_tid, &r_addr, &l);
        if (packet) {
            printf("(tid %d from node %d) >> ", r_tid, r_addr);
            print_sample_data(packet, l);
            free((void *)packet);
        }
    }
    return 0;
}

void print_sample_data(uint8_t *data, int len) {
    attr_t *attr = (attr_t *)data;
    uint8_t *dataptr = (uint8_t *)attr->value;
    int i, tmp;
    int num_samples = attr->length/2;

    printf("[%d] #samples=%d, channel=%d: \n",
            data_pkts_read, num_samples, attr->type&0x03);

    for (i = 0; i < num_samples; i++) {
        // MSB first for MDA4X0 sample data
        tmp = (dataptr[2*i] << 8) + (dataptr[2*i + 1]);
        printf("%d ", tmp);
    }
    putchar('\n');
}
```

Complete code for an structural data acquisition system on

Tenet

Tenet: An Architecture for Tiered Sensor Networks



(Distributed) computing substrate
Application development happens here

Simplify this interface!
- sensor-addressable tasking
- (reliable) data collection
- *generic* signal processing
  - this can evolve

Programmable data acquisition

Still need duty cycling, MAC, localization,timesynch, routing, congestion control *etc.* here

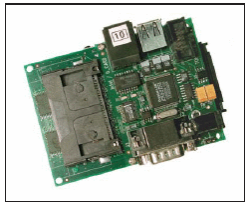
Joint work with Deborah Estrin and Eddie Kohler

How Tenet Works

```
int main(int argc, char **argv)
{
    unsigned char b[200];
    int l, tid = 0;
    uint16_t r_tid, r_addr;

    /* construct a task packet */
    l = construct_task(b, task_string);
    if (l < 0) {
        printf("task description error!! \n");
        exit(1);
    }

    /* send the task */
    tid = send_task((uint16_t)tid, l, b);
    if (tid > 0) {
        printf("\ntask has been sent with tid %d!!\n", tid);
    } else {
        exit(1);
    }

    /* receive response packets */
    while(1) {
        unsigned char *packet = receive_packet(&r_tid, &r_addr, &l);
        if (packet) {
            printf("(tid %d from node %d) >> ", r_tid, r_addr);
            print_sample_data(packet, l);
            free((void *)packet);
        }
    }
```

Tenet Code

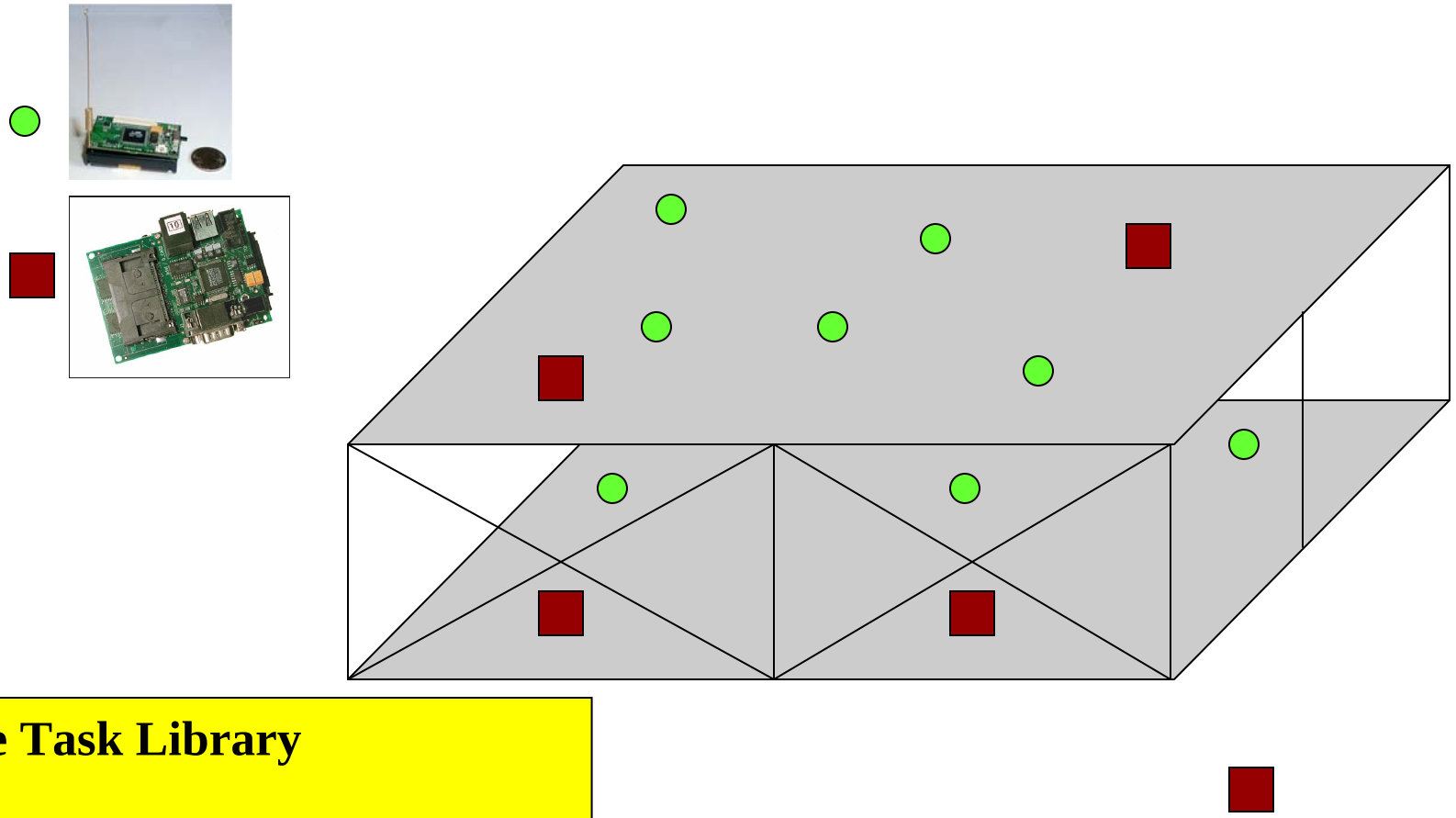
In Tenet, applications run on an upper-tier node



They **task** lower-tier nodes: linear data-flow tasking language

Application collects and correlates observations

How Tenet Works

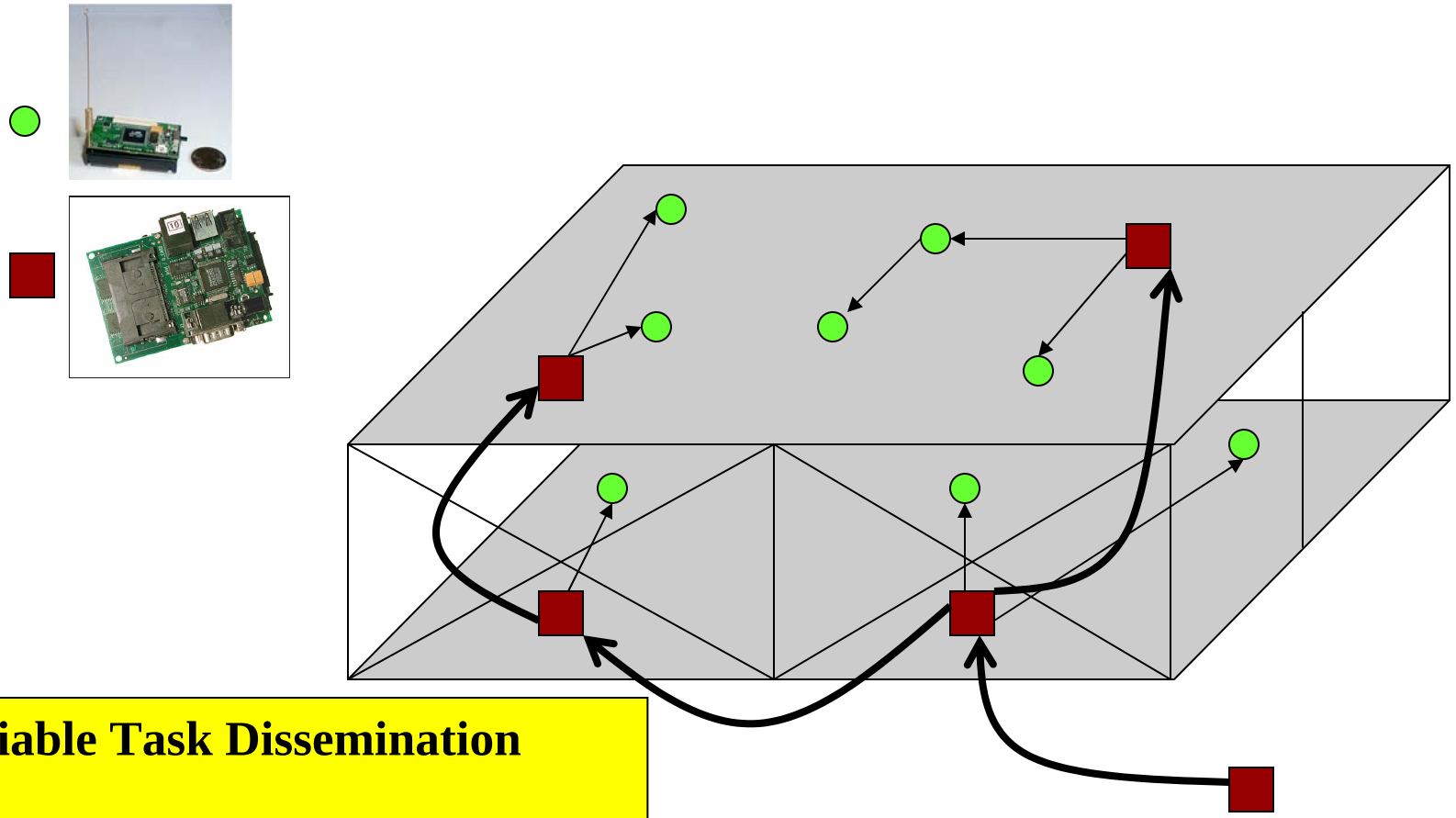


The Task Library

- A simple linear data-flow language and pre-defined processing elements
- Tasks executed on sensors

sampleMDA(…) -> DetectOnset() -> sendStream()

How Tenet Works

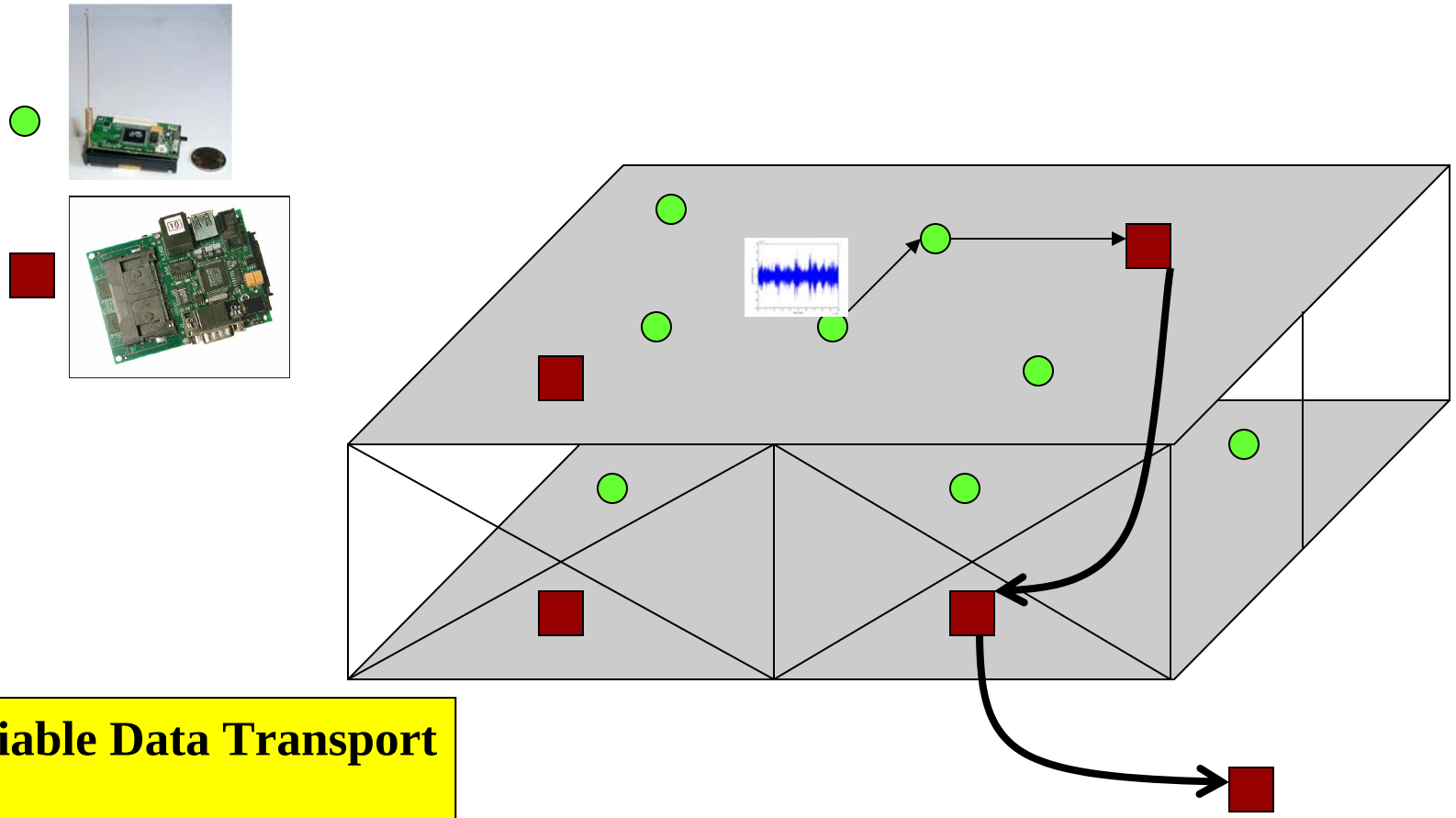


Reliable Task Dissemination

- Uses ideas from epidemic propagation and gossiping
- Extraordinarily robust

sampleMDA(…) -> DetectOnset() -> sendStream()

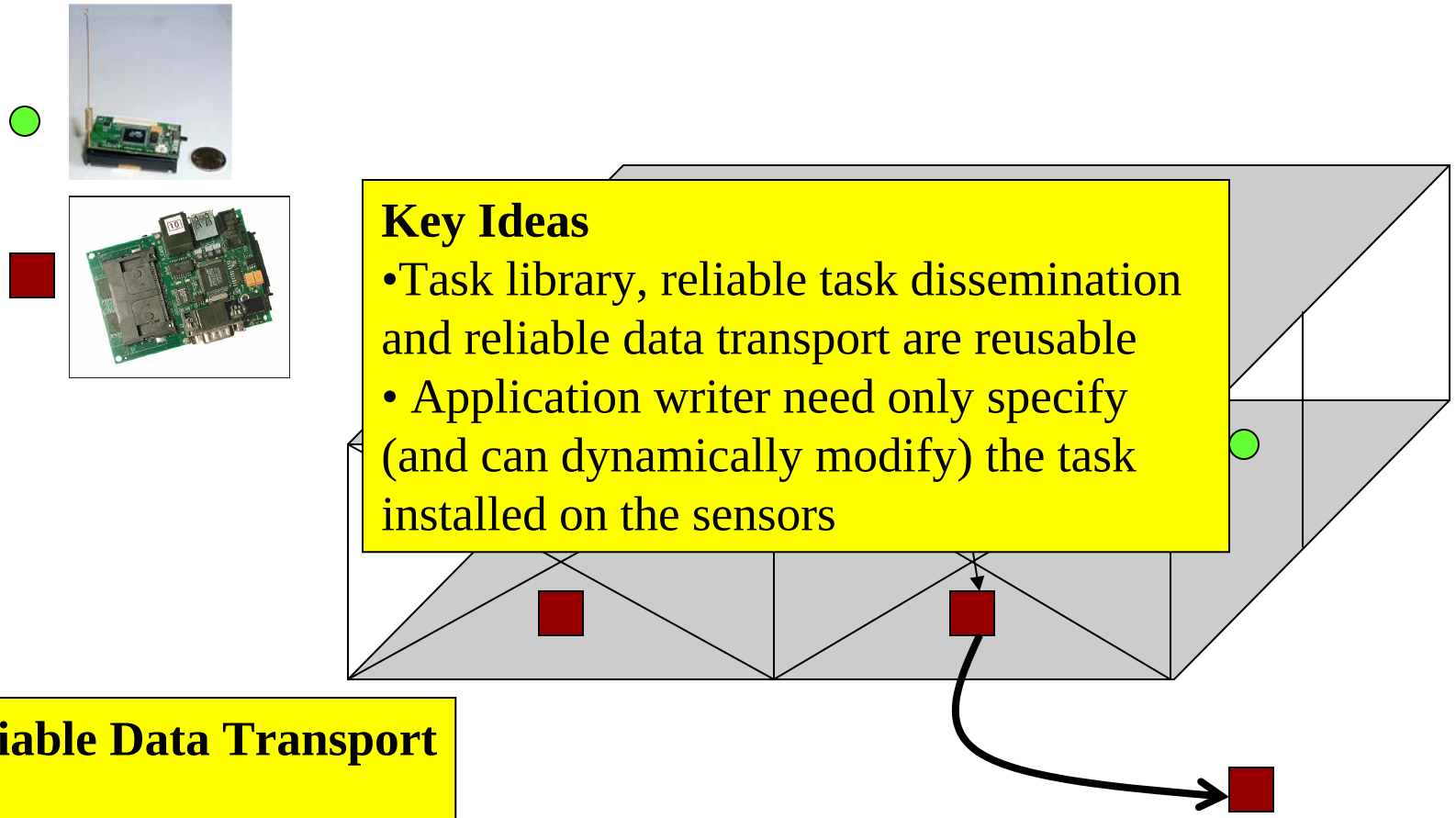
How Tenet Works



Reliable Data Transport

- Similar to Internet's TCP
- Retransmits lost data

How Tenet Works



Key Ideas

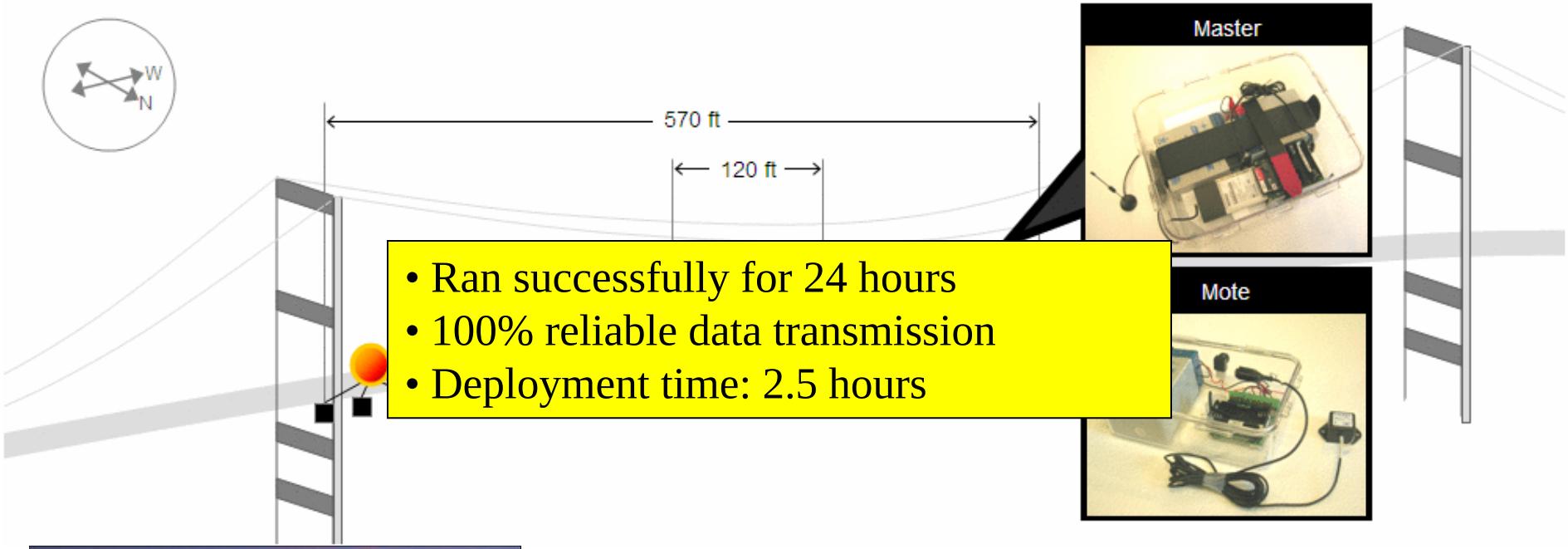
•Task library, reliable task dissemination and reliable data transport are reusable

• Application writer need only specify (and can dynamically modify) the task installed on the sensors

Reliable Data Transport

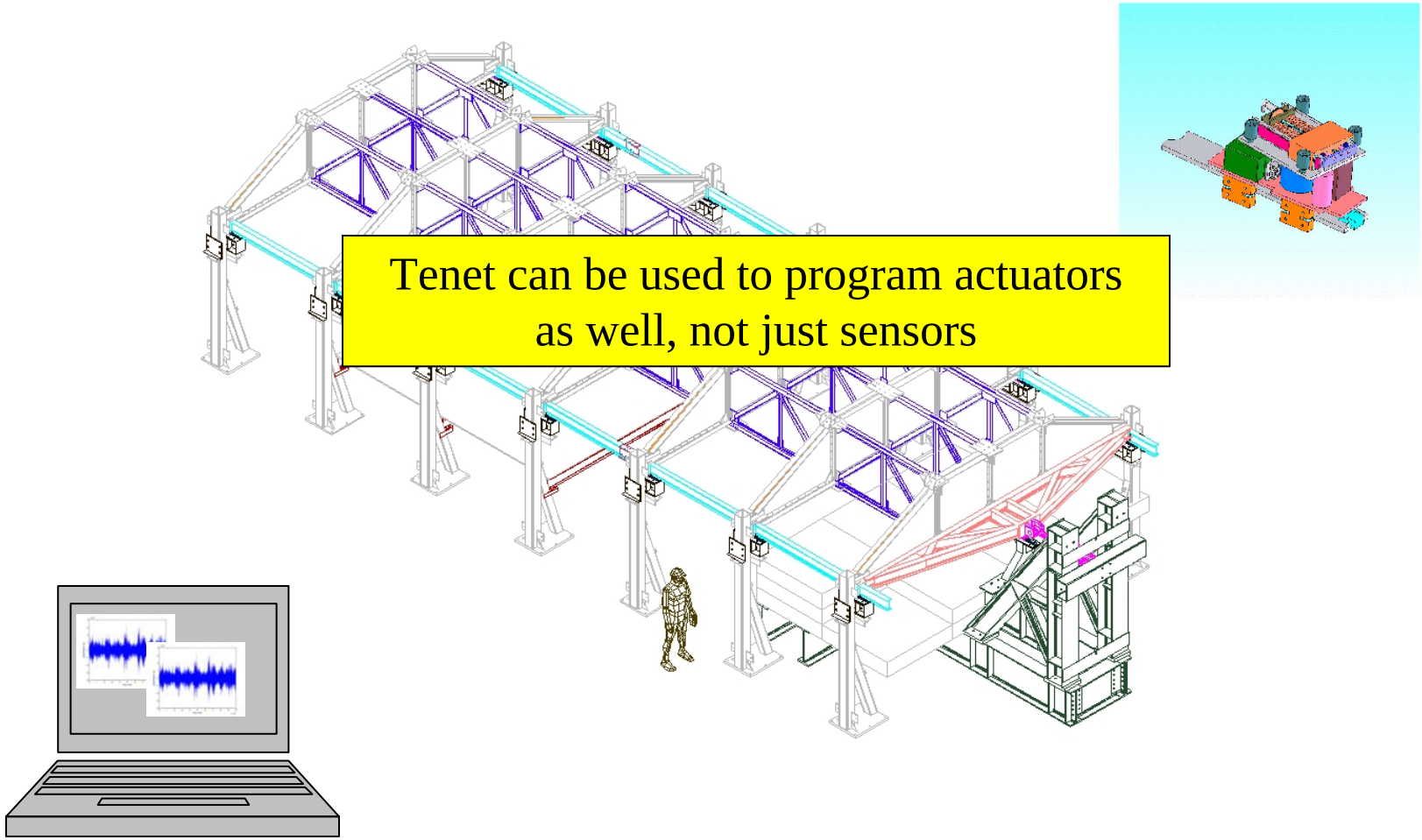
• Similar to Internet's TCP
• Retransmits lost data

Continuous Monitoring of the Vincent Thomas Bridge using Tenet



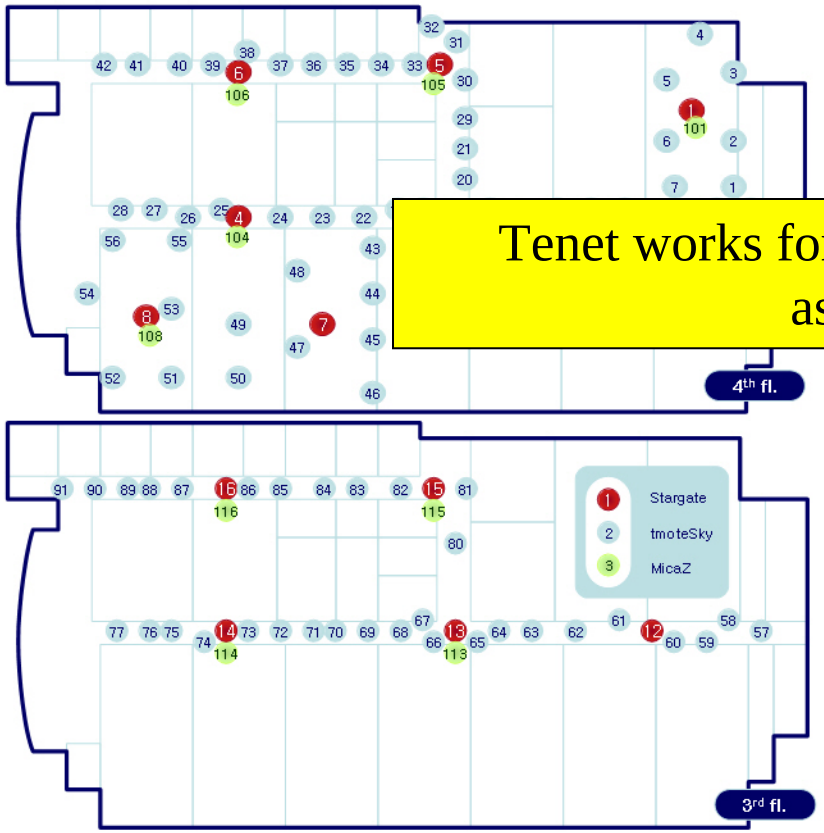
- Ran successfully for 24 hours
- 100% reliable data transmission
- Deployment time: 2.5 hours

Automated Structural Testing Using Tenet



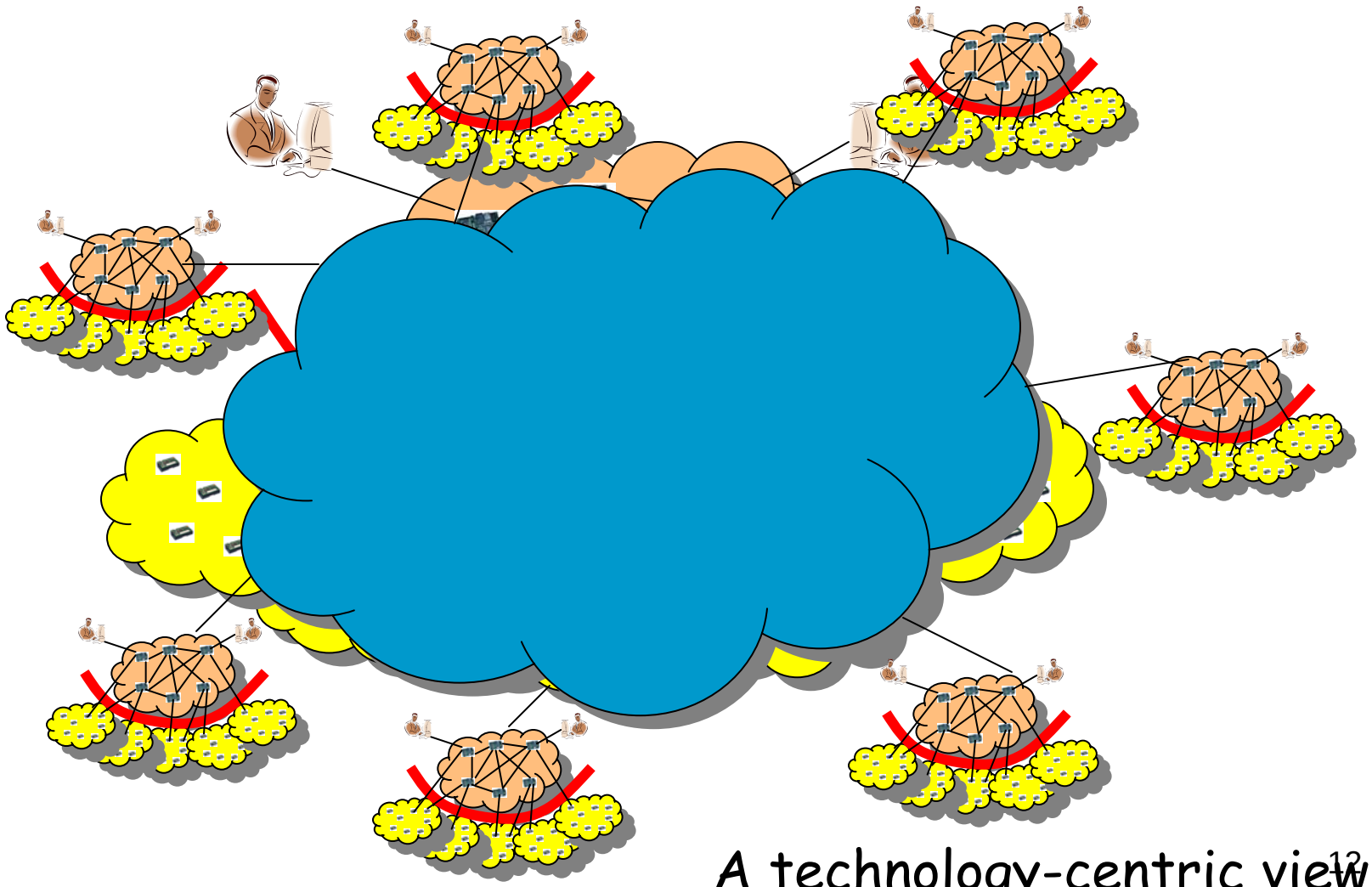
Tenet can be used to program actuators as well, not just sensors

Pursuer-Evader Games using Tenet



Tenet works for other applications as well

Where do we go next?



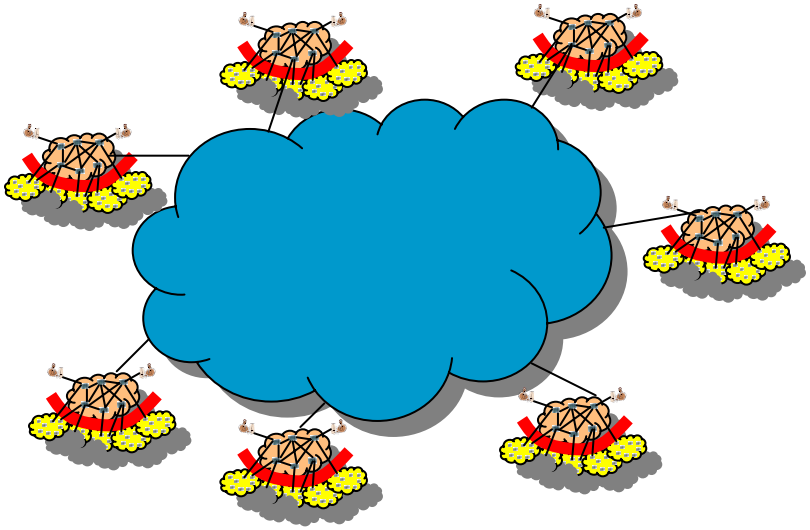
A technology-centric view

Where do we go next?

Visualize, monitor, **control,**
and manage all activities, in near real-time,
in a metropolis

A user-centric view

The Challenge



Infrastructure for Reactive Sensing Systems

- How do we "tie" together disparate sensor nets?
  - What (distributed?) software systems do we put in place to build these large-scale reactive sensing systems?
- Is this a solved problem?
  - "We know how to do search, DHTs, DNS"
  - Reactivity at short-time scales is an important constraint
  - Canonical application: controlling the power-grid

What Abstractions Should We Provide?

- Search
  - Flexible, a very natural paradigm
  - … but can it be sufficiently reactive for (partially) automated control?
    - i.e. does the implementation have to be necessarily centralized
- Structured access
  - Prior work on data-centric and database methods
  - Amenable to decentralized implementation
  - Real-time
- Physical access
  - "Assembler" equivalent: provide direct physical access to sensors or nets
- In reality, will need a combination of these

Roadblocks

- Security
  - But can leverage solutions developed for other distributed infrastructures
- Privacy!
  - This is critical for the kind of federated sensor nets we're talking about
- Having real applications to play with

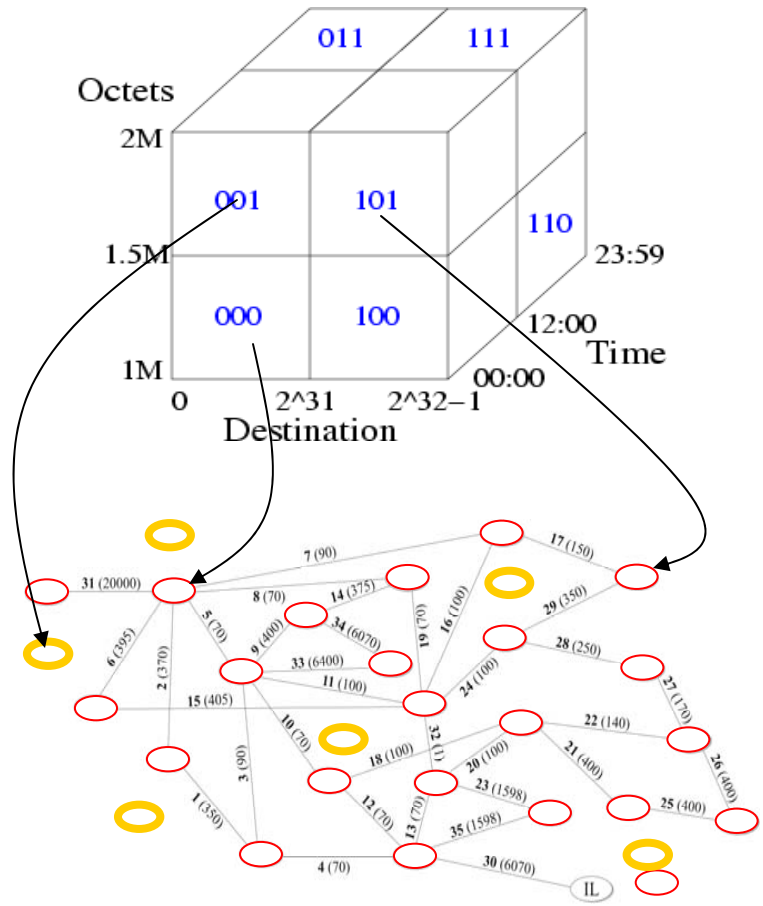
Leverage Point

- Intriguing similarity with wide-area network monitoring
- Examples:
  - data-cube queries for Internet traffic records
  - anomaly detection

Multi-Dimensional Range Queries

Monitoring Suspicious Network Traffic

- Functionality
  - Supports creation of multi-dimensional indices using an Internet overlay
- Applications
  - Traffic monitoring:
    - Finding suspicious flows
    - Performance monitoring
  - Routing problem root-cause analysis
  - More generally, correlating events from different sources
- Design elements
  - Hypercube overlay
  - Proximal hashing of data
  - Data-space to hypercube mapping based on data distributions

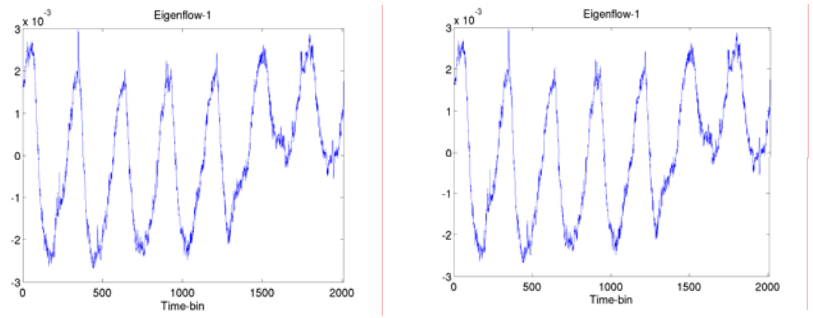


Joint work with Christophe Diot and Gianluca Iannaconne

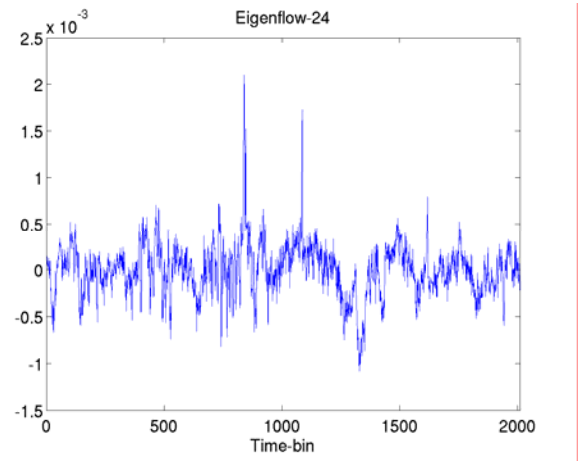
Anomaly Detection

Looking for Needles

- Goal
  - To detect IP flows that are "anomalous" (unusual traffic volumes, number of ports *etc.*)
- Approach
  - Dimensionality reduction techniques
  - Apply principal components analysis on a sketch of the traffic seen at a router
    - relies on an empirical observation that most Internet traffic is well-described by a few components
  - Sketches are additive, so the technique is amenable to distribution



Sketching preserves normal components



Anomalies reside in the abnormal subspace

Joint work with Mark Crovella, Christophe Diot and Gianluca Iannaconne