

# TIME-EACM

## A Transport Information Monitoring Environment: Event Architecture and Context Management

Alastair R. Beresford

University of Cambridge

11th July 2006

# The transport problem

Current transport infrastructure in the UK is failing to support its users:

- ▶ Road congestion in the UK costs of the order of 20bn per annum
- ▶ 85% of senior business people believe that investment decisions are influenced by the quality of transport
- ▶ Nearly 15% of trains in the UK arrived late in 2005-06
- ▶ Many Cambridge streets continue to exceed air quality measures for various pollutants



# Can technology help?

Hypothesis:

*investment in monitoring, distribution and processing of traffic information will cause a substantial and significant increase in transport efficiency*

If true, then we can:

- ▶ improve business efficiency
- ▶ increase social cohesion and inclusion through better public transport
- ▶ reduce pollution

# TIME-EACM

## Goal:

*to investigate, design and build a secure yet open interface to support the controlled sharing of transport-related data*

## In more detail:

- ▶ Develop, build and deploy sensors
- ▶ Construct an event-based middleware that:
  - ▶ hides low-level sensor aggregation from applications
  - ▶ integrates high-level context models with query support
  - ▶ provides open and collaborative access to data
  - ▶ ensures privacy of personally identifiable information
  - ▶ controls distribution of commercially-sensitive data
- ▶ Write real-world applications that fully exploit the architecture
- ▶ Evaluate all of the above
- ▶ Focus on solutions for the City of Cambridge

# Application scenarios

- ▶ Car park status via SMS
- ▶ Interactive bus arrival signs at bus stops
- ▶ Taxi booking and dispatch via the web
- ▶ Estimated journey times for car drivers
- ▶ Inter-modal journey planner
- ▶ Congestion detection, prediction and avoidance
- ▶ Behaviour inference to support long-term policy planning
- ▶ ...

# Why an open architecture?

We have several of these applications already, so why an open architecture?

- ▶ Applications are easier to build with such support
  - ▶ encourages quicker innovation and deployment
  - ▶ reduces total cost for a given set of applications
- ▶ Enables market place in capturing, processing and distribution of transport data
- ▶ Integration and inference from several sources becomes possible
  - ▶ enables better applications through richer context data

# What have we achieved so far?

Started work in many of the key areas of the grant; so far:

- ▶ Road sensors: scoot, **Irisys**, sentient van
- ▶ Environment: city pollution data, weather station
- ▶ Inference: **road topology**, mean vehicle speed  
**energy-aware, in-network processing**
- ▶ Long-term strategy: privacy-aware congestion charging
- ▶ **Middleware**: distributed, standards-based design & build underway

## Sensor example: infrared movement tracking

- ▶ Utilises an infrared 16x16 pixel CCD to track movement over the field-of-view
- ▶ Normally used in indoor environments as a people counter
- ▶ We have deployed this sensor outdoors to increase our coverage area
  - ▶ Initial deployment reveals success rate of 80-90%
- ▶ Sensor preserves privacy: individuals cannot be identified

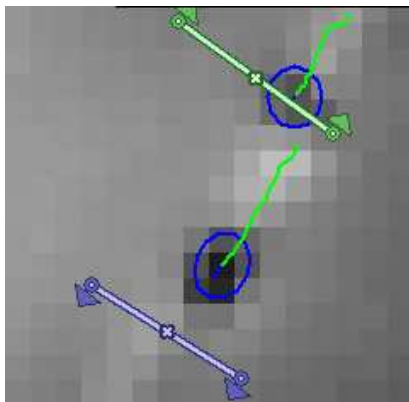


*(Collaborative work with Irisys Ltd.)*

## Sensor example: infrared movement tracking



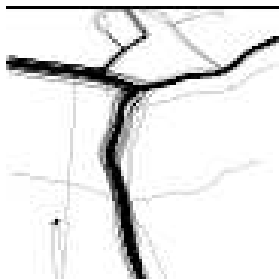
video



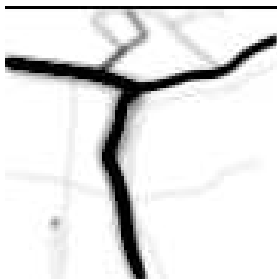
sensor

*(Collaborative work with Irisys Ltd.)*

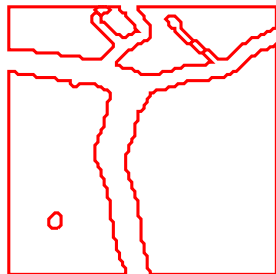
## Inference example: using GPS traces to update maps



collate data



convolution filter

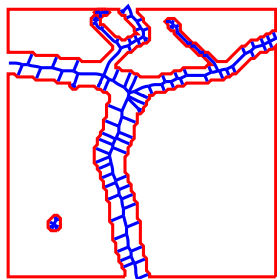


contour follow

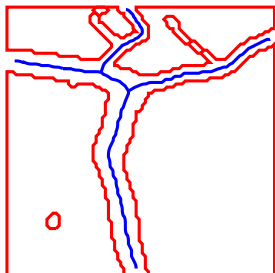
- ▶ Collect GPS traces from private vehicles
- ▶ Use traces to infer information about the environment
- ▶ Traditional techniques: photos, council DBs, probe vehicles

*(Joint work with Jonathan Davies and Andy Hopper)*

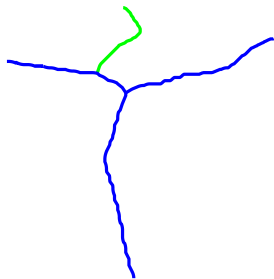
## Inference example: using GPS traces to update maps



Voronoi graph



segment removal



directionalise

- ▶ Enables up-to-date maps & other data besides
- ▶ How do we collect vast quantities of data?
- ▶ Where does the processing occur?

*(Joint work with Jonathan Davies and Andy Hopper)*



# Sensor data management

Sensor node constraints:

- ▶ Limited bandwidth, energy and processing power

Objective:

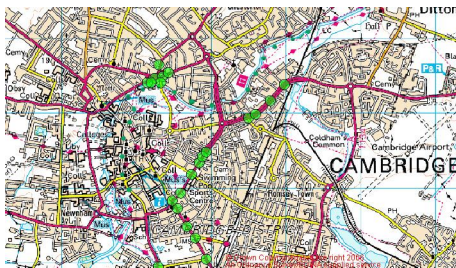
- ▶ Reduce the volume of data propagated from the sensors to the middleware

Approach:

- ▶ Push data processing into the network.
- ▶ Use lossy compression techniques that exploit:
  - ▶ temporal correlations; and
  - ▶ spatial correlations of time series data

*(Work done at Birkbeck by N. Trigoni, A. Guitton and A. Skordylis)*

# SCOOT traffic data



- ▶ 160 inductive loops deployed in Cambridge
- ▶ flow and occupancy readings
- ▶ information generated every 5 minutes
- ▶ data recorded 6 months

*(Work done at Birkbeck by N. Trigoni, A. Guitton and A. Skordylis)*

# Proposed framework

## Compression and clustering:

- ▶ Apply Fourier or wavelet transforms to compress the data generated at each sensor
- ▶ Cluster sensors together based on the degree of similarity between their compressed data

## Data transmission to the middleware:

- ▶ Transmit the compressed data of only one node per cluster (the cluster-head)
- ▶ For the remaining nodes of the cluster, transmit regression parameters that correlate their data to that of the cluster-head

*(Work done at Birkbeck by N. Trigoni, A. Guitton and A. Skordylis)*

## Middleware—design features & goals

- ▶ Supports data fusion
- ▶ Distributes current and historical data
- ▶ Hotplug support for new components and applications
- ▶ Privacy & security protection
- ▶ Support for maintenance
- ▶ Robust to failure
- ▶ Explicitly measures uncertainty of data
- ▶ Amenable to performance measurement
- ▶ Mathematically predictable performance

## Middleware—current implementation

- ▶ Two types of components: stream and storage
- ▶ No central component (well, almost)
- ▶ Events distributed to listeners by stream component in XML
- ▶ Historical queries to storage component via XML/RPC
- ▶ XML schemas used to enforce correct data formats
- ▶ TCP/IP used to connect together different components

## Middleware—resource discovery service

- ▶ An example of a storage component
- ▶ Resource discovery runs on a well known IP/port
- ▶ One standardised XML “meta” schema for all storage components
- ▶ Accessed via XML/RPC
- ▶ Actions: list, get, create, update, delete
- ▶ Public key crypto used to support distributed access control
- ▶ Stores location (IP/port) of each component; apps can query RDS
- ▶ Assists with component migration

## More information

<http://www.cl.cam.ac.uk/Research/TIME>