Hourglass:
A Stream-Based Overlay Network for Sensor Applications

Peter Pietzuch, Jeff Shneidman, Jonathan Ledlie
Matt Welsh, Margo Seltzer, Mema Roussopoulos

Division of Engineering and Applied Sciences
Harvard University

Emergency Medical Care

Sensor support for emergency medical care

- **Motes** attached to patients collect vital signs (pulse ox, heart rate, ...)
- **EKG Mote** with PDA runs on Windows CE platform
- PDAs carried by EMTs receive vital signs and enter field reports
- Ambulance correlates with patient records at hospital

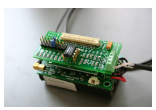
Characteristics:

- Many heterogenous patient sensors act as data sources
- Real-time streamed data
- Partial network connectivity to ambulance

Volcano Monitoring

Logging seismic activity of a volcano (Tungurahua) in Ecuador

- Sensors record low-frequency infra-sound (5 Hz)
- Survey physical structure of the inside of a volcano
- Many sensors ⇒ **Mountain tomography**



Goals:

- Satellite uplink from base station at volcano
- Push queries into sensor network
- Fuse data from several volcanoes
- Collaboration between universities

Application Features

Large number of distributed data sources

- Dynamic, heterogeneous sources with imperfect network connectivity

In-network, real-time processing of data

- Aggregation close to sources
- Efficient resource utilisation

Multiple applications sharing sensor networks

- Different administrative domains with custom security policies

⇒ Need for a reusable and efficient Internet infrastructure for data collection and processing

- ▹ *Scalable, distributed, fault-tolerant implementation*
- ▹ *Sophisticated optimisation strategies*
- ▹ *Fast deployment of novel applications*

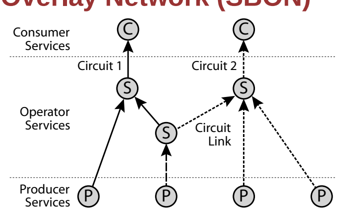
Stream-based Overlay Network (SBON)

Main Components:

- **Consumer** (C)
- **Producer** (P)
- **Service** (S)
- **Circuit**

Pinned and *unpinned* services

Hourglass prototype



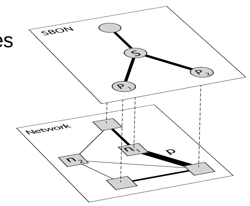
Placement Problem

Need to *place* unpinned services

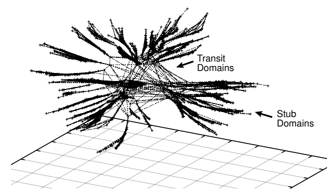
- Placements have costs
- Optimisation problem
- Need approximation without global knowledge

Global and *application* costs

- Network utilisation
- Application delay penalty



Latency Space



Solve placement problem in virtual space

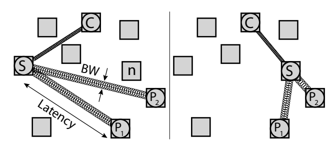
- Euclidean distance = latency
- 1550-node transit-stub topology
- Efficient encoding of global topology knowledge
- Scalable implementation

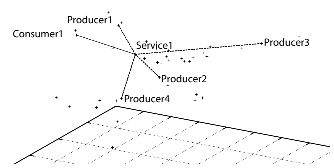
Relaxation Placement

Model circuits as network of springs

- Spring extension = latency
- Spring constant = bandwidth

Minimises network traffic





Map solution back to physical space

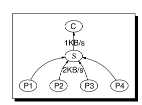
Advantages:

- Decentralised implementation
- Supports cross-circuit optimisation

Evaluation

6 Placement algorithms in simulator:

- **Optimal**: exhaustive search
- **Relaxation**
- **IP Multicast**: place at M/C routers
- **Producer/Consumer**: place at P/C
- **Random**: random placement

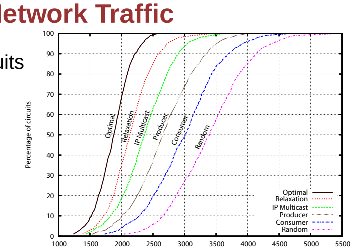


Network Traffic

1000 4-producer circuits in simulator

Overhead traffic:

Optimal	1.00
Relaxation	1.15
IP Multicast	1.27
Producer	1.43
Consumer	1.60
Random	1.81



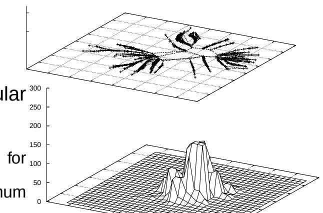
Node Load

Distribution of service placement

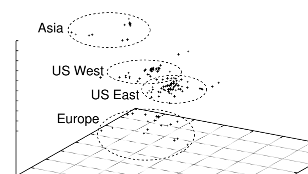
- Load-balancing?

Transit domains more popular for service placement

- Traffic goes there anyway
- Enable transit domains for service placement
- Need a cap on maximum number of placed services



PlanetLab



Verified simulator results on PlanetLab

- Distributed test-bed with 439 nodes
- PlanetLab topology in latency space

Physical topology of PlanetLab unknown

- *Scriptroute* supports remote traceroute

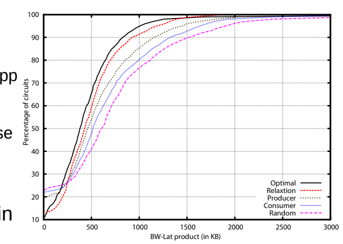
Used simulator for placement decision

- Deployed *Hourglass*, a stream-processing app

Results:

- Network utilisation close to optimal
- App delay penalty remains low

Distribution of traffic in transit on PlanetLab



Future Work

Fully-decentralised implementation on PlanetLab

- Adaptable to network dynamics and circuit evolution
- Convergence results of distributed relaxation

Explore potential of cross-circuit optimisation

- Investigate circuits used by realistic applications
- Large-scale circuit optimisation (reuse services)

Summary

SBONs enable future sensor applications

- Service placement is a crucial problem in SBONs
- Efficient resource utilisation and network awareness is important

Relaxation placement

- Spring relaxation model in latency space
- Scalable decentralised implementation with low comms. overhead
- Supports cross-circuit optimisation

Experiments

- Transit-stub topology: Relaxation is close to optimal; better than IP M/C
- PlanetLab: Verified results in shared networking environment

hourglass@eecs.harvard.edu

http://www.eecs.harvard.edu/∼syrah/hourglass