Voxnet: a Rapidly Deployable Distributed Acoustic Sensor Network

Lewis Girod

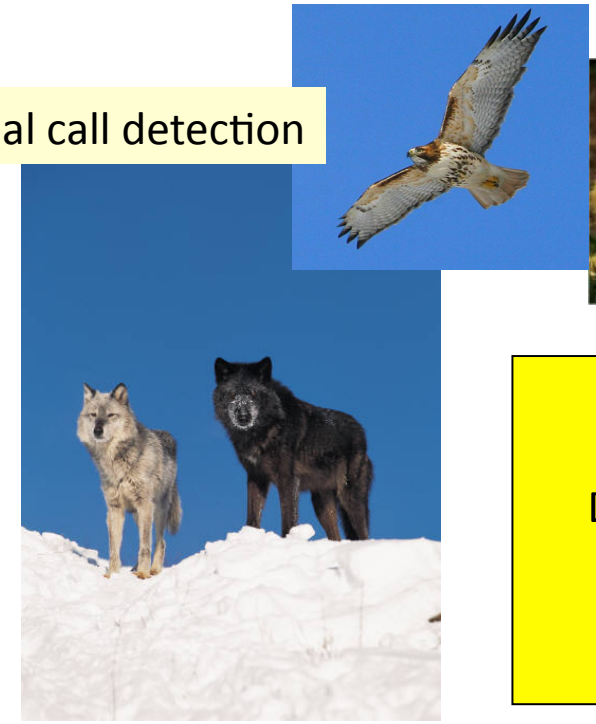
4 April 2016

6.S062

This project was the work of many collaborators from UCLA, MIT, and Coventry (UK), and was supported by several grants from the National Science Foundation.

Distributed Acoustic Sensing Applications

Animal call detection

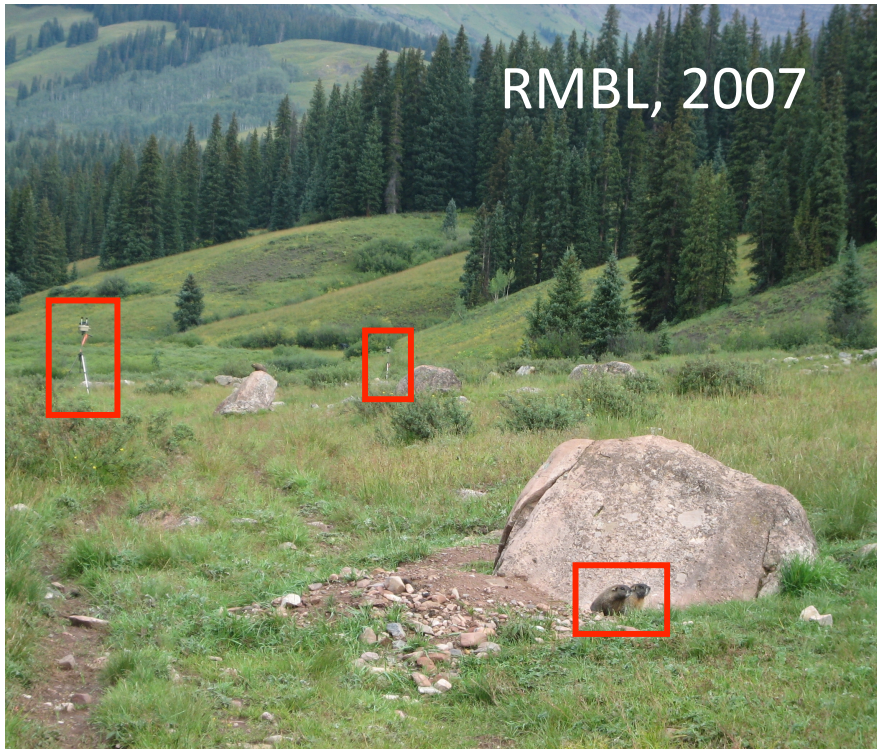


Condition-based maintenance

Goal:
Develop a *platform* to support distributed acoustic sensing

Perimeter security, Vehicle tracking

VoxNet



Terrestrial bio-acoustics, e.g.,

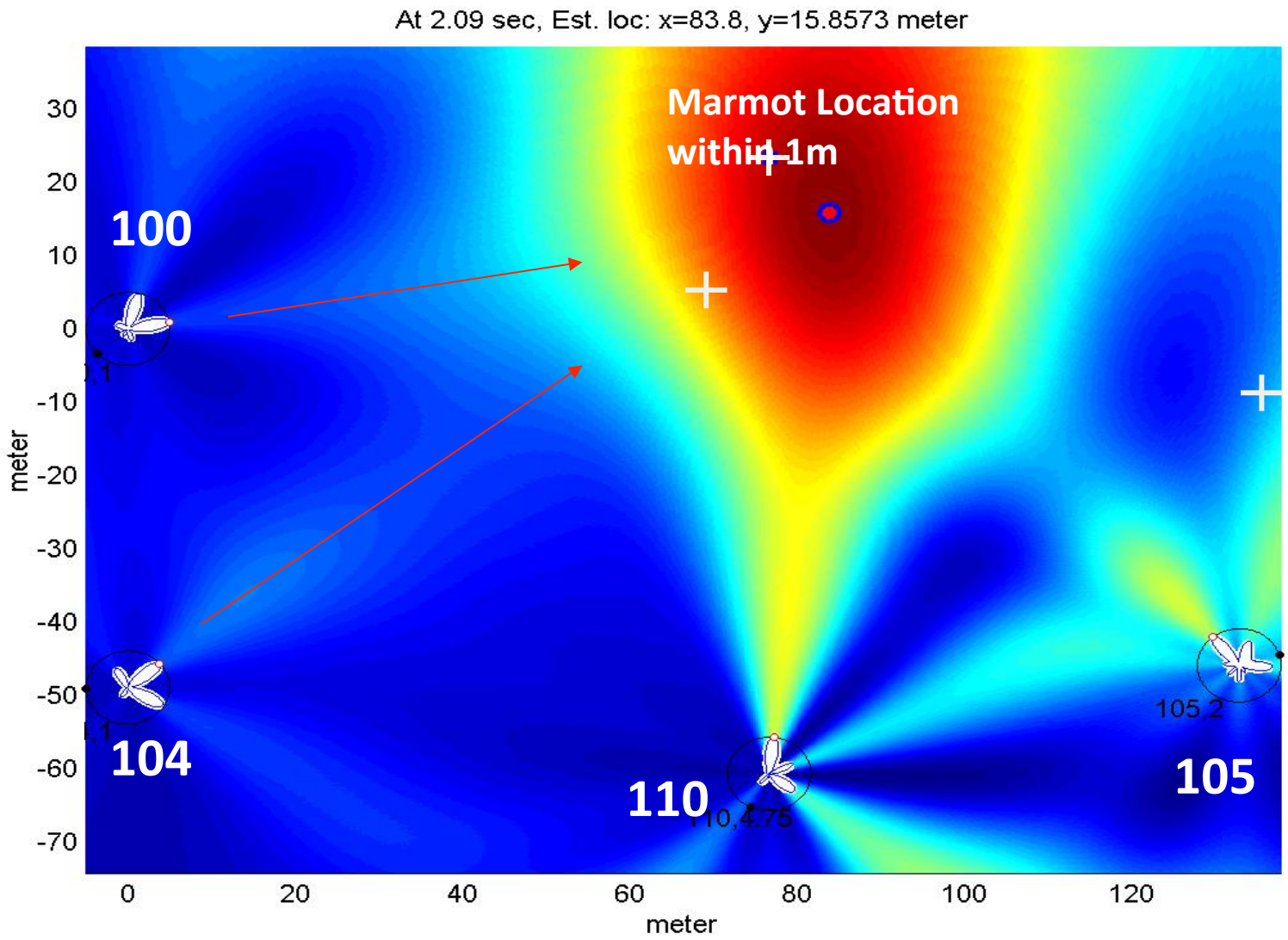
- marmot alarm calls
- bird calls
- wolves
- human encroachment

Allen, et al., VoxNet: An Interactive, Rapidly-Deployable Acoustic Monitoring Platform, IPSN 2008

Initial Applications and Deployments

- Rocky Mountain Biological Lab
  Gothic, CO
  - Marmot localization study
  - 2 weeks in 2006
  - 3 months in 2007
- Chajul Biological Station
  Chiapas, MX
  - Bird localization and classification
  - 2 weeks in 2007

Call localization application



Ali, et. al., An Empirical Study of Collaborative Acoustic Source Localization, IPSN 2007.

Key Challenges in Terrestrial Acoustics

- "Free space" is rare
  - Blocked LOS: foliage, earth
  - Reverb, multi-path, attenuation
- Spectrum cluttered with noise
  - Wind, rivers, rain
  - Other animals
- Wireless performance
  - GPS often blocked by foliage
  - 2.4 GHz absorption

Design Implications

Difficult environment →

- Need many vantage points
- Need sensors closer to sources

→ Distributed →

- Greater complexity – more parts
- More ways to be mis-configured

→ Deployable →

- Wireless, self-contained, resilient
- Self-configuring, easy to maintain

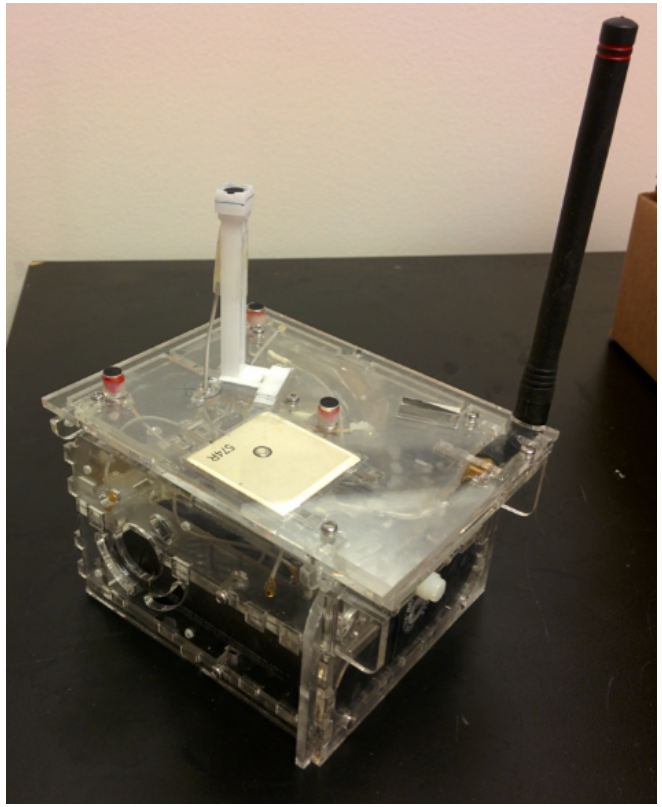
VoxNet Platform

Hardware:

OMAP3
256MB RAM
SD Card
BT/802.11B
433 Mhz
Supervisor uC
Li+ battery
Charge controller

Sensors:

4x48KHz audio
GPS
Internal temp/RH



Key Features:

Rapid deployment
- Small / light
- Self-contained

Self calibration
- Absolute location
- Precise relative location
- Precise orientation
- Time synchronization

App environment
- Field programmable
- Real time responses
- Spill to disk

Hardware Platform Evolution

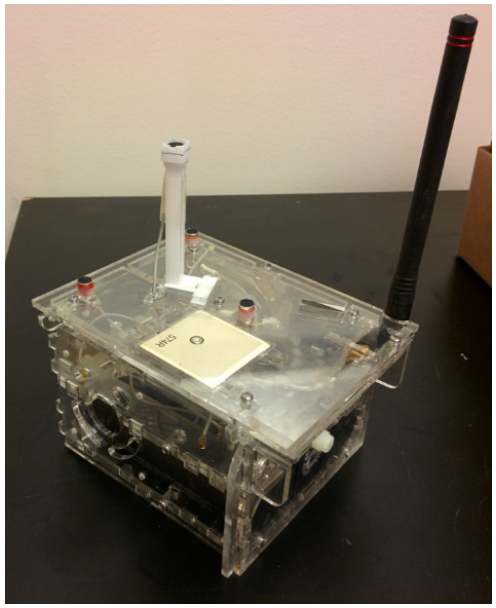
Version 1 (2005)

Large pelican box (16x10”)
External gel cell battery
Separate mic head

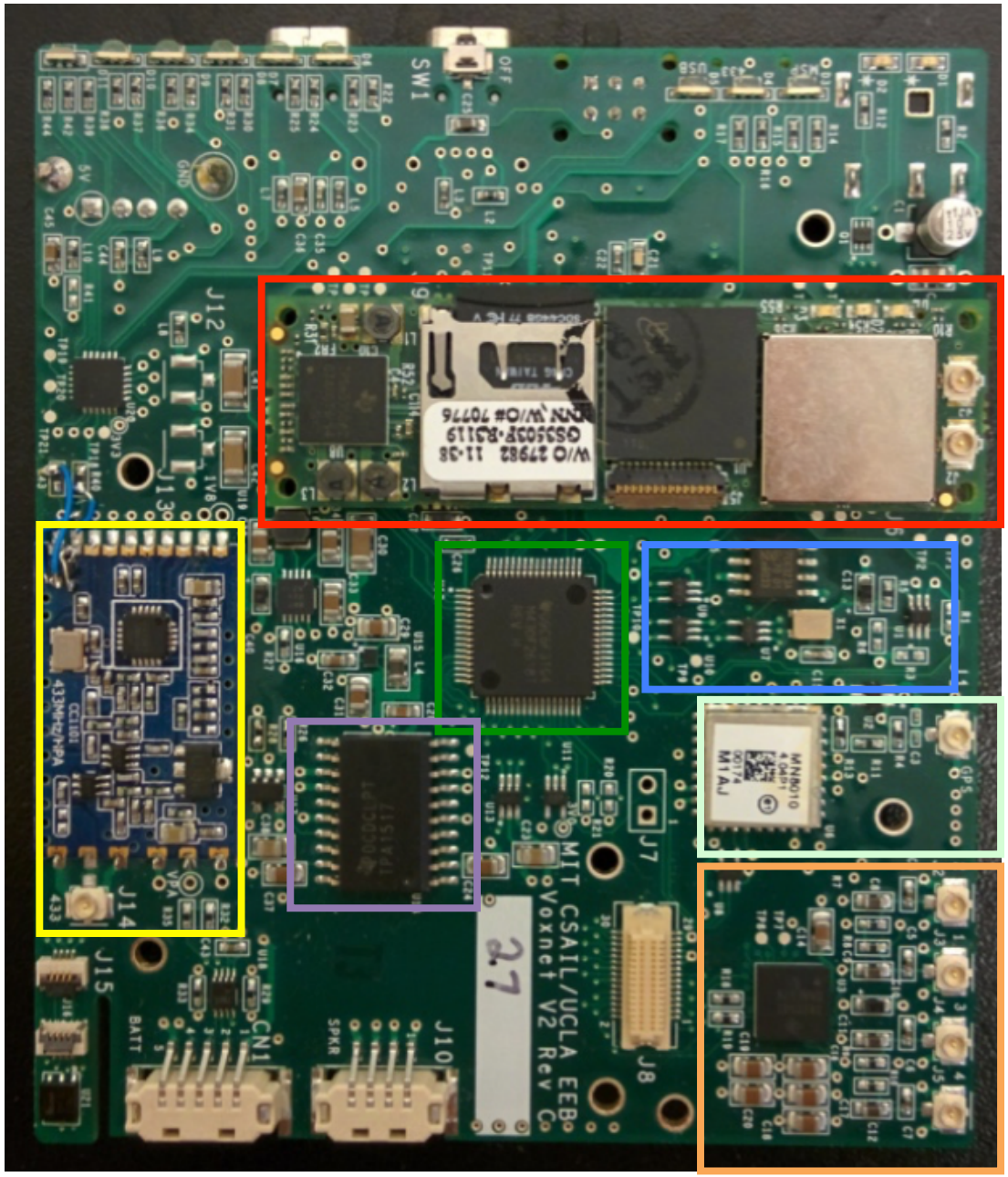
Version 2 (2007)

Small pelican box (8x6”)
Internal Li+ battery
Detachable mics and speaker cube.

Version 3 (2009)



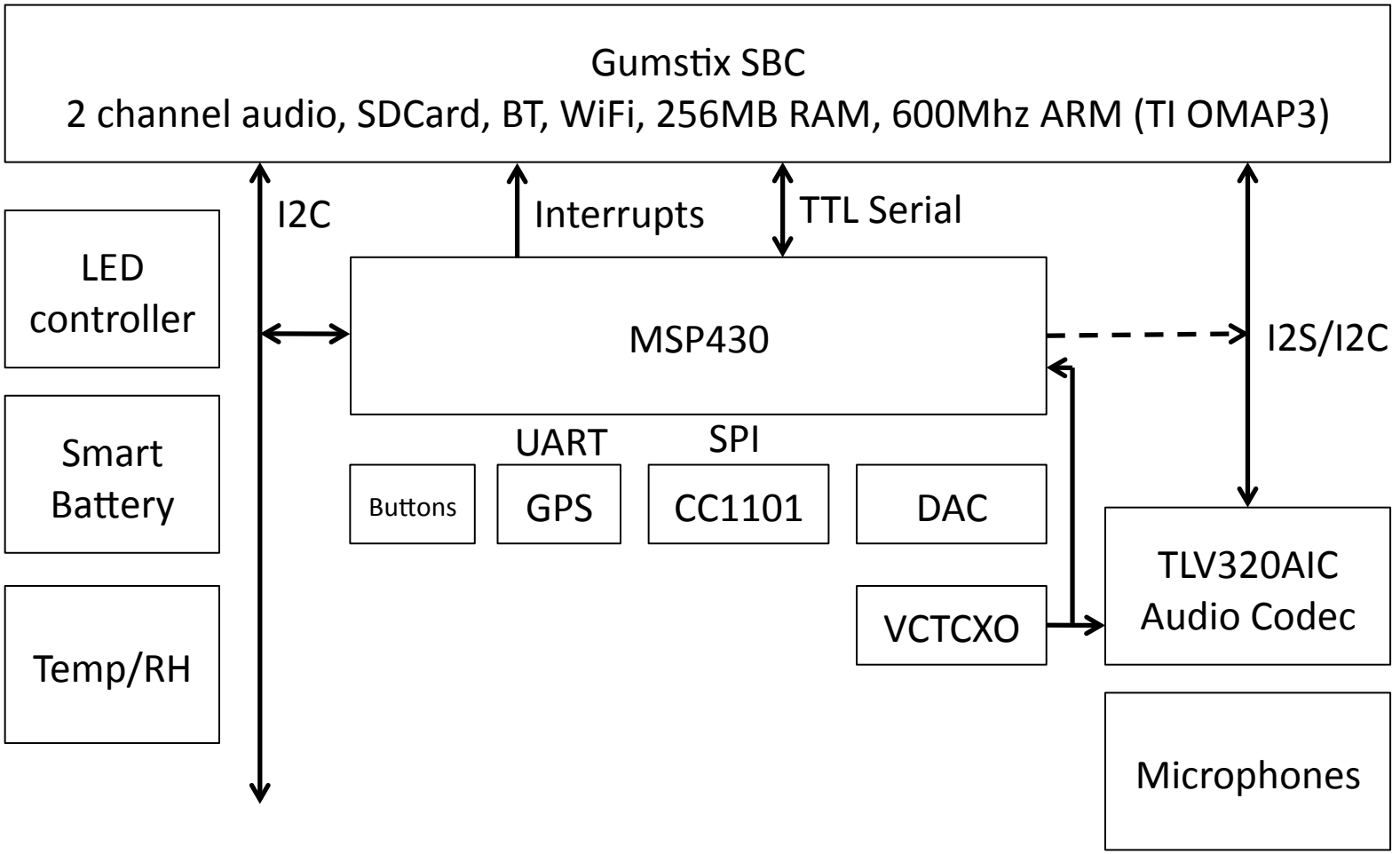
Laser-cut case (5x4”)
Internal Li+ battery
Integrated mics and speakers. Designed as a peripheral board for an off the shelf SBC.



Hardware

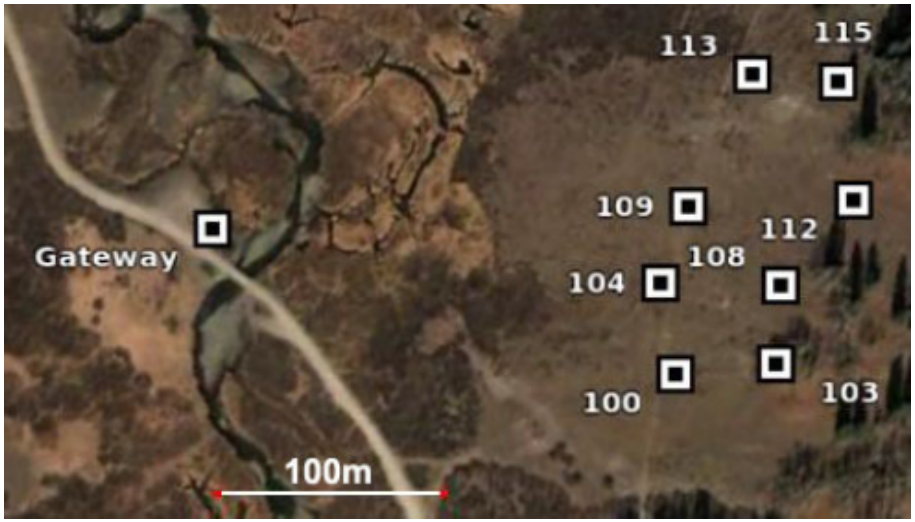
- Gumstix SBC
- CC1101 module and P/A
- MSP430 controller
- Frequency locked audio clock
- Audio P/A
- GPS
- 4 Channel codec
- Not visible: Power supplies, battery charger, LED controller

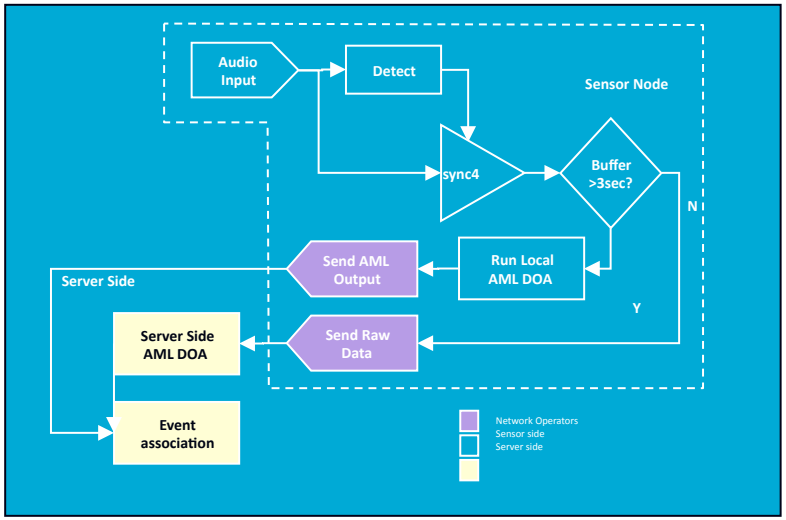
Hardware Block Diagram



VoxNet Software Platform

- Maintains a multi-hop wireless network
- Time synchronization between nodes
- Precise relative location and orientation
  - Does not rely on GPS
  - Much higher accuracy than GPS
- Voxnet programming interface

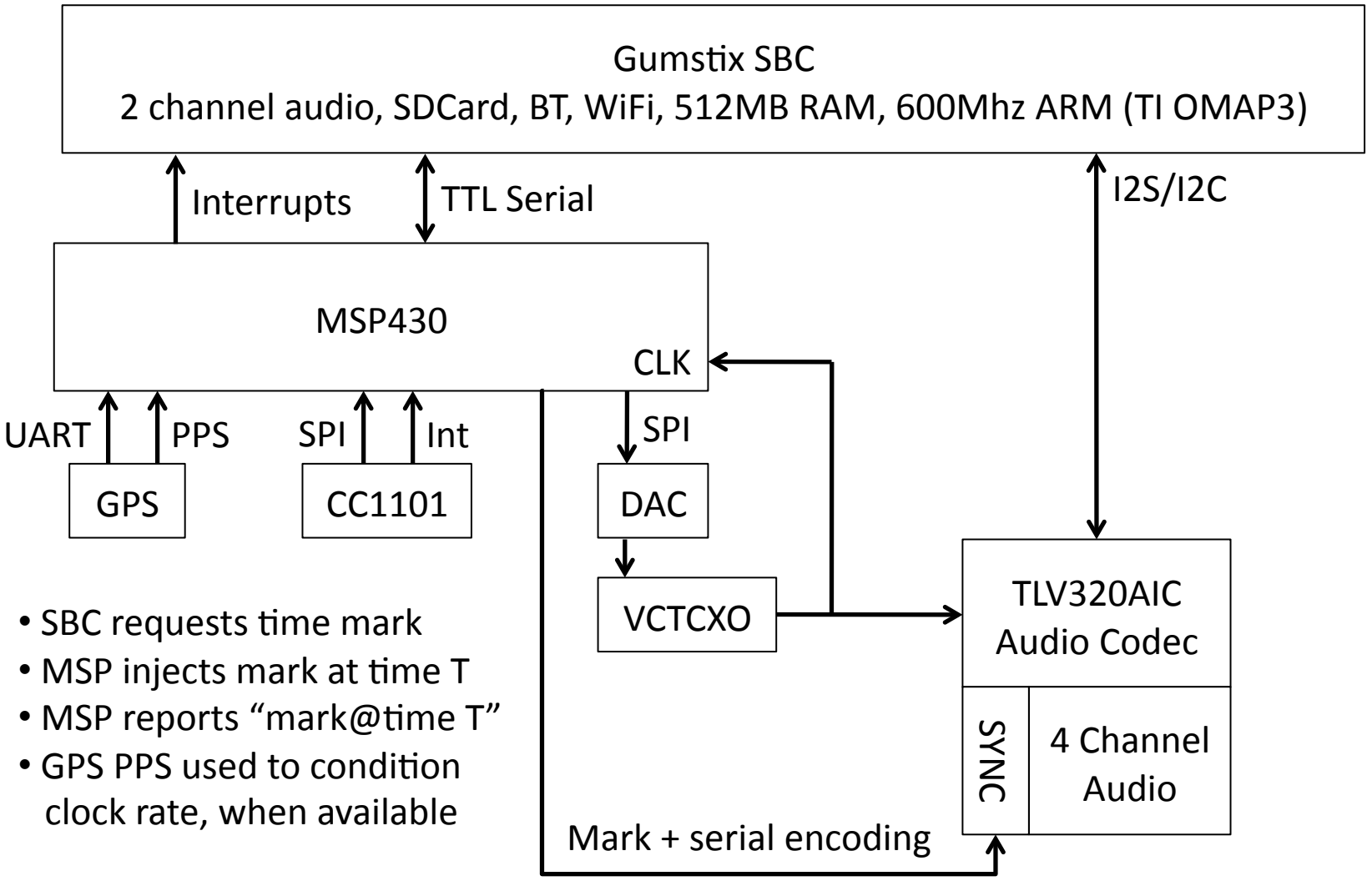




Time Synchronized Sampling

- Why?
  - Substantially reduces error in distributed TDOA
    - E.g. time of flight audio range estimation
  - Post-facto analysis of long-running distributed recordings
- Implementations requires multi-layer support
  - Hardware, Software, Network
- Principle of free-running clock
  - Let clock run, rather than jumping it
  - Maintain and record mapping/conversion data

Hardware support for time-synchronized audio sampling

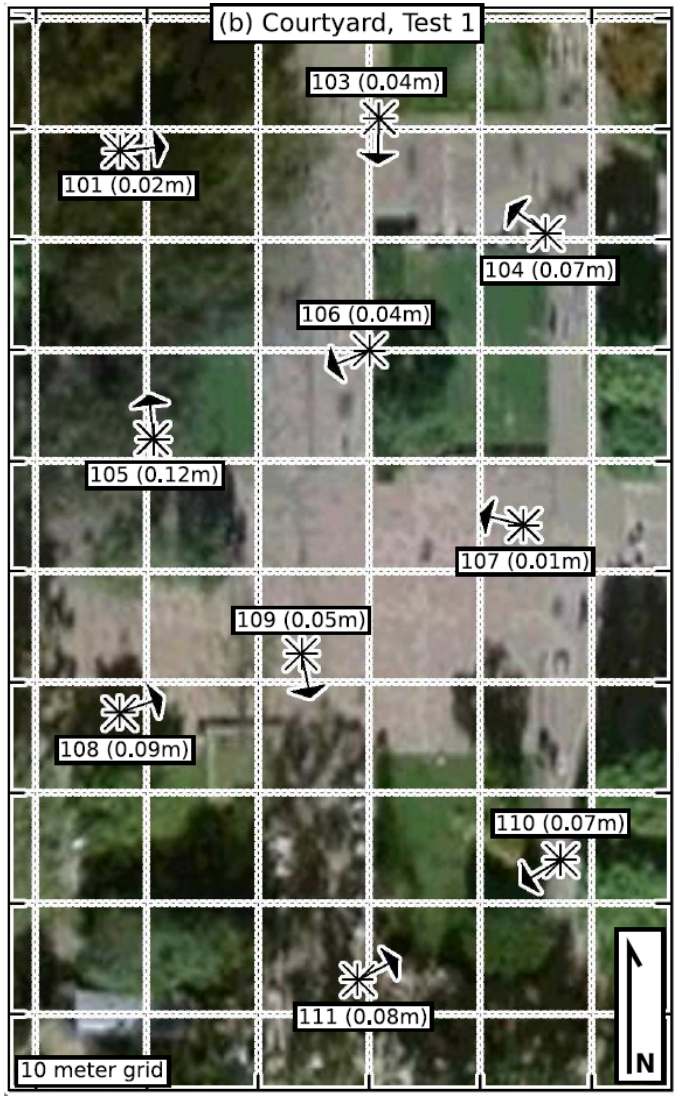


Software/Network support for TS

- At least one node with access to GPS
  - Declares itself ‘level 0’ and propagates GPS time in sync packet
- Each node maintains an up-to-date mapping between its internal clock and GPS time
  - Non level-0 nodes transmit best mapping as level n+1
- Live analysis uses current mapping
- Historical mapping recorded for use with recorded data

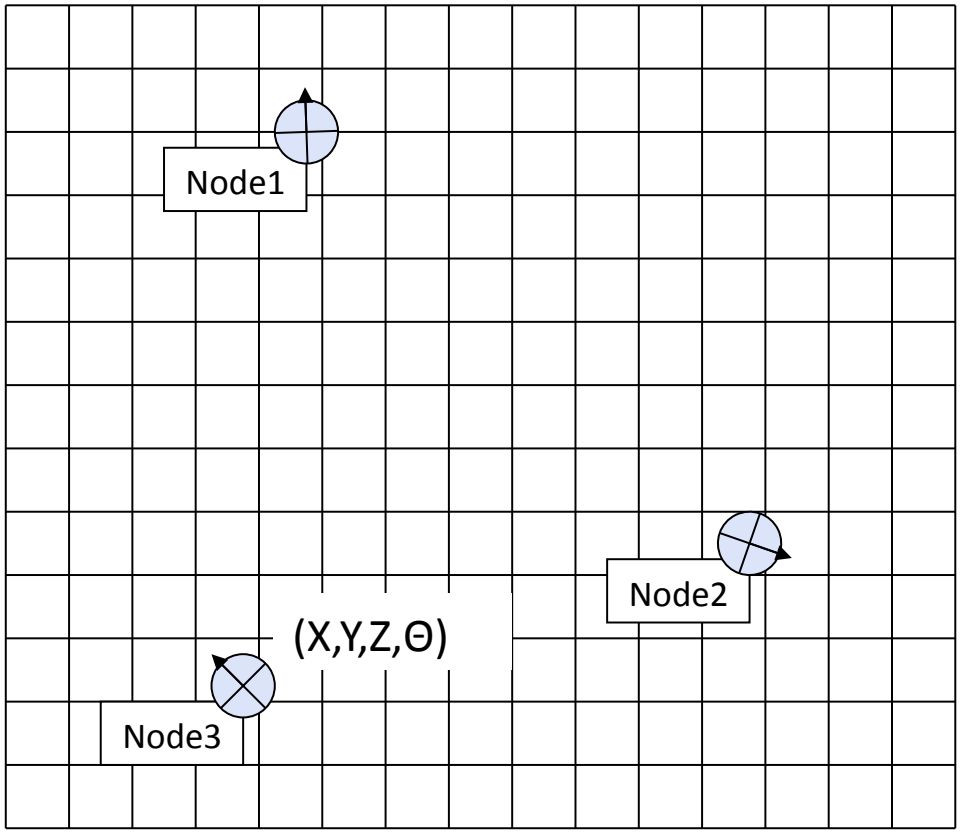
Precise location and orientation

- Based on audio TOF, TDOA
  - Over-constrained system of non-linear constraints
- Highly accurate
  - 5-10 cm position error in 200 m (0.05%)
  - 1 degree orientation error
- Works well in foliage
  - Redundant measurements ensure accuracy



Girod, et. al., The Design and Implementation of a Self-calibrating Acoustic Sensing Platform, SenSys 2006

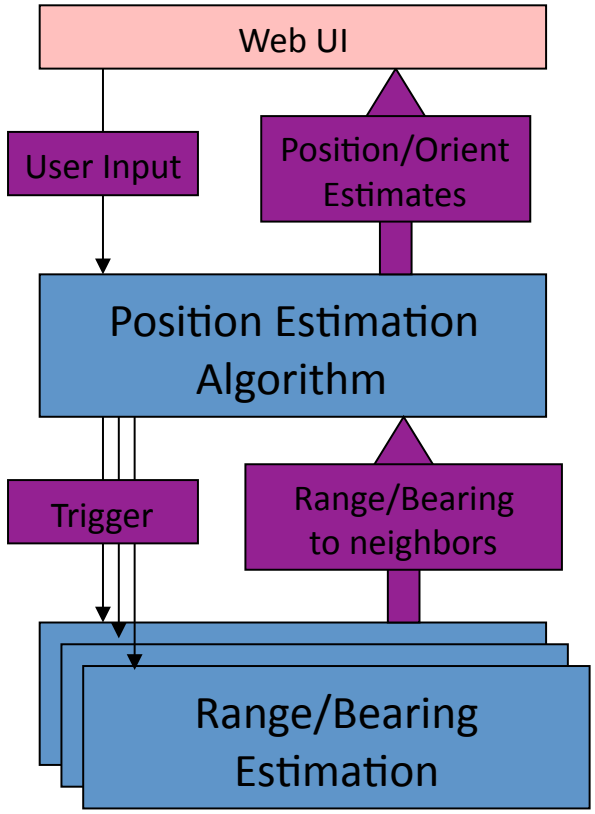
Self-calibration:
Precise location and orientation



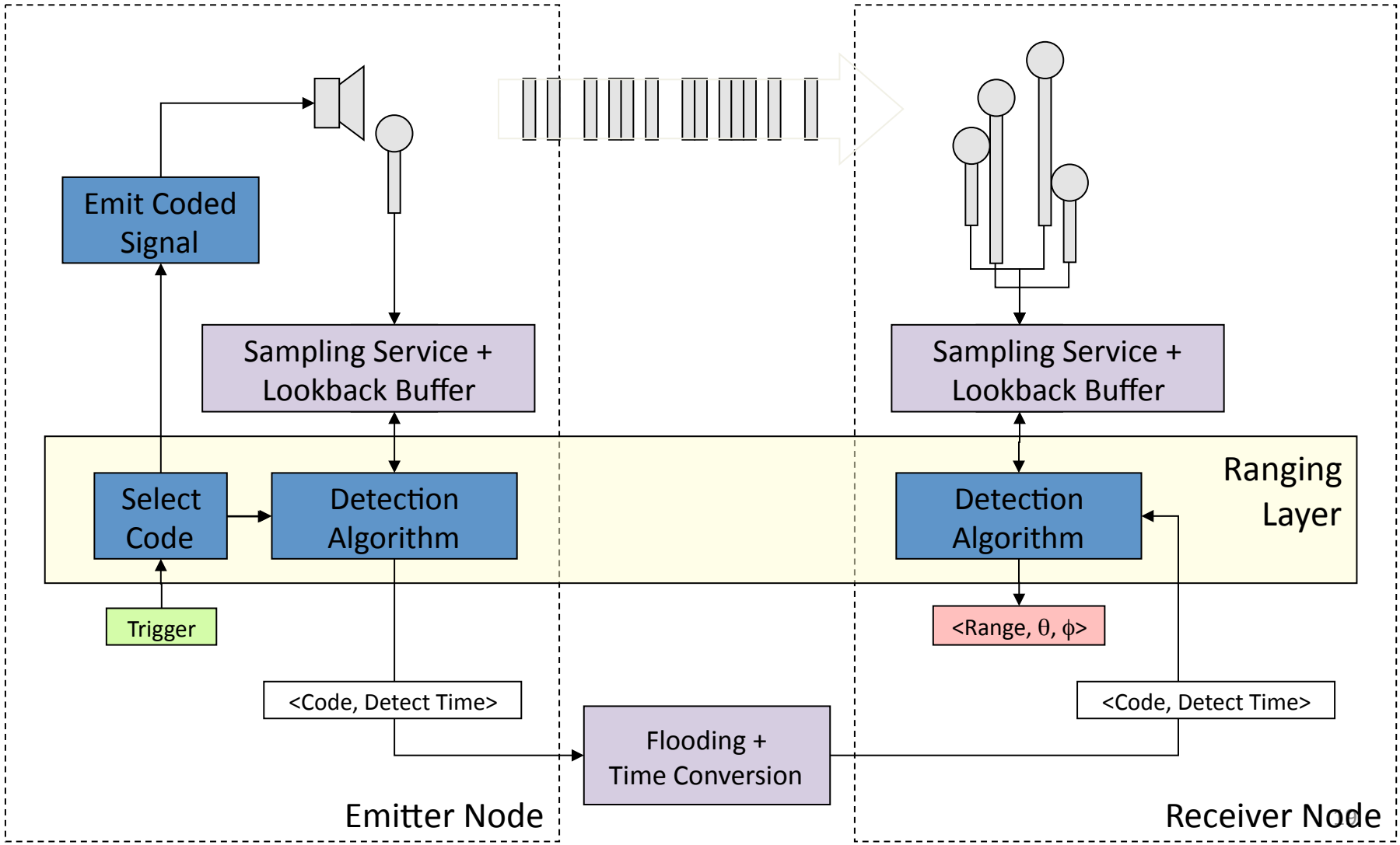
- For each node, estimate:
  - 3D position (X,Y,Z)
  - Array orientation Θ
- Fit relative coordinates to survey and/or selective GPS availability
- Assume all arrays are level
- Metric (given ground truth):
  - Avg 2D, 3D position error
  - Avg orientation error

System Overview

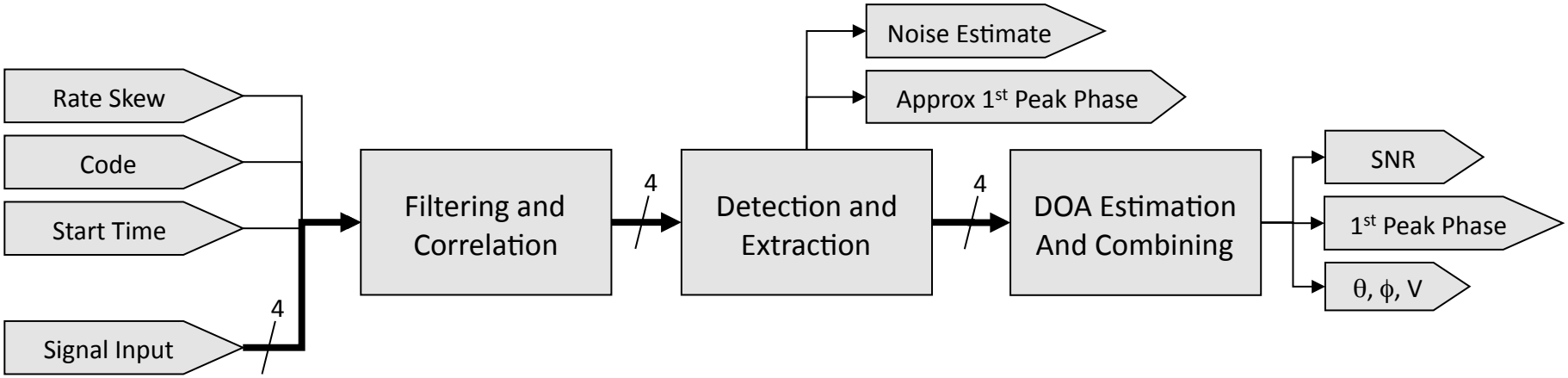
- Range/Bearing Estimation
  - Distributed through network
  - Emitter chirps, other nodes detect phase of chirp via correlation
  - Estimate AOA using local TDOA
  - Report (R,θ) data to position layer
- Position Estimation
  - Centralized algorithm
  - Triggers each node to chirp in turn, receives (R,θ) estimates
  - Least squares algorithm to compute relative map (X,Y,Z,Θ)
  - Fit map to surveyed locations



Range and Bearing Estimation System Overview



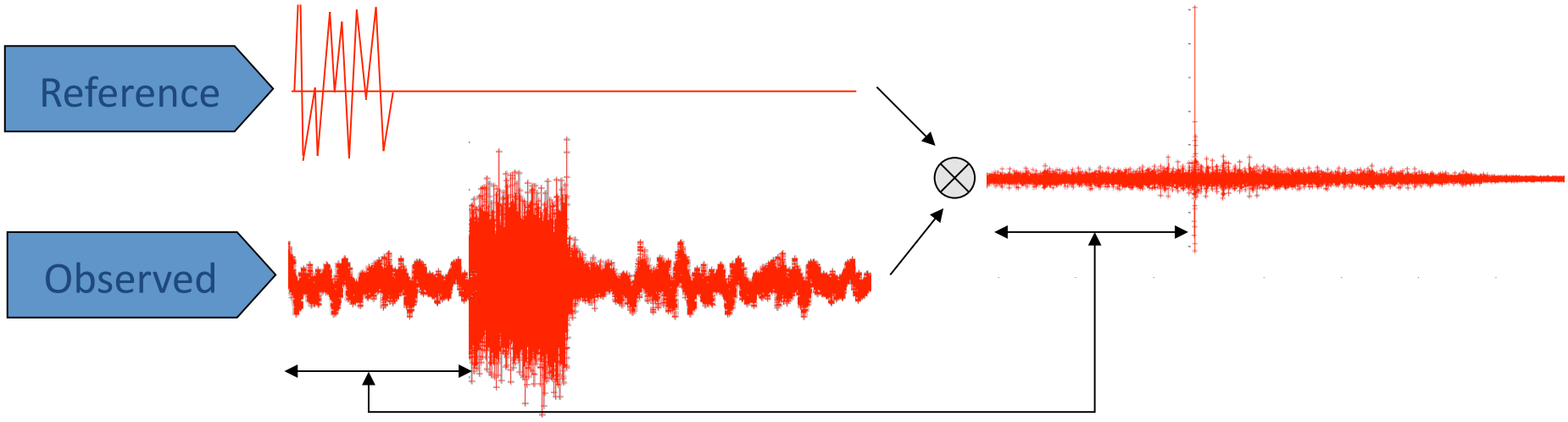
Range and Bearing Estimation Algorithm



- Inputs:
  - The input signals from the microphones
  - The time the signal was emitted (used to select from input signal)
  - The PN code index used
- Outputs
  - Peak phase (i.e. range): R
  - The 3-D direction of arrival: θ, ϕ
  - Signal to Noise Ratio (SNR)

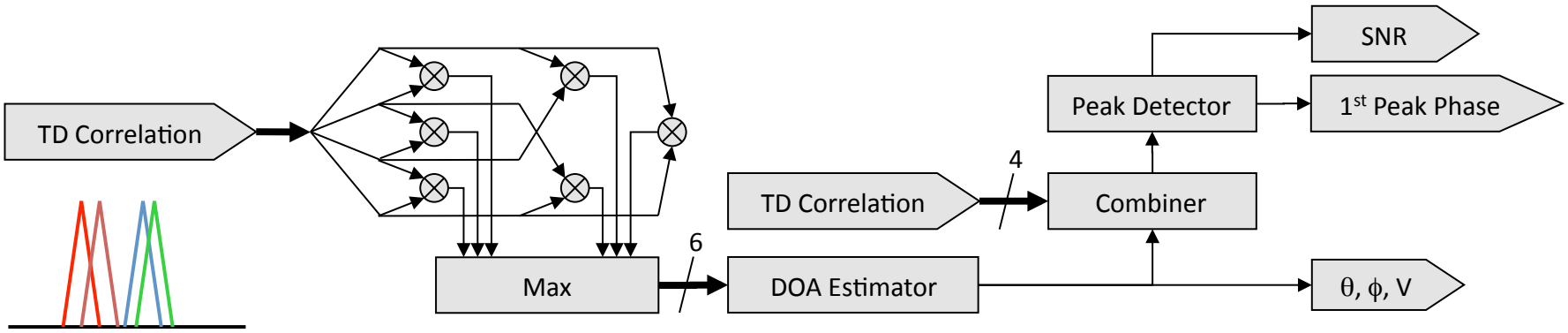
Correlation

- Signal detection via “matched filter” constructed from PN code
  - Observed signal S is convolved with the reference signal
  - Peaks in resulting “correlation function” correspond to arrivals
  - Earliest peak is most direct path



Lag = Time of Flight

DOA Estimation and Combining Stage

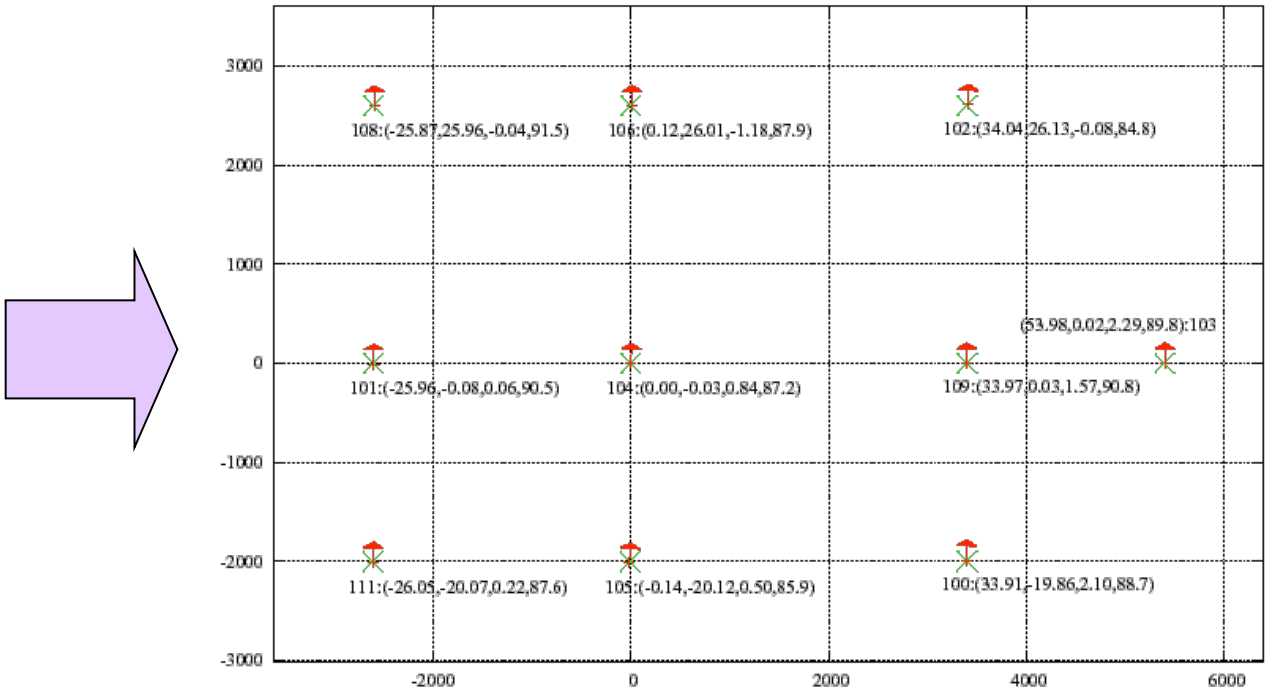


- 6-way cross-correlation of "peak region" → DOA Estimator
  - DOA Estimator: non-linear least squares fit of phase offsets to array geometry, determine most likely DOA
  - Resilient to perturbations in microphone placement
- DOA estimate used to recombine signals to improve SNR
- Final peak detection yields final range estimate R

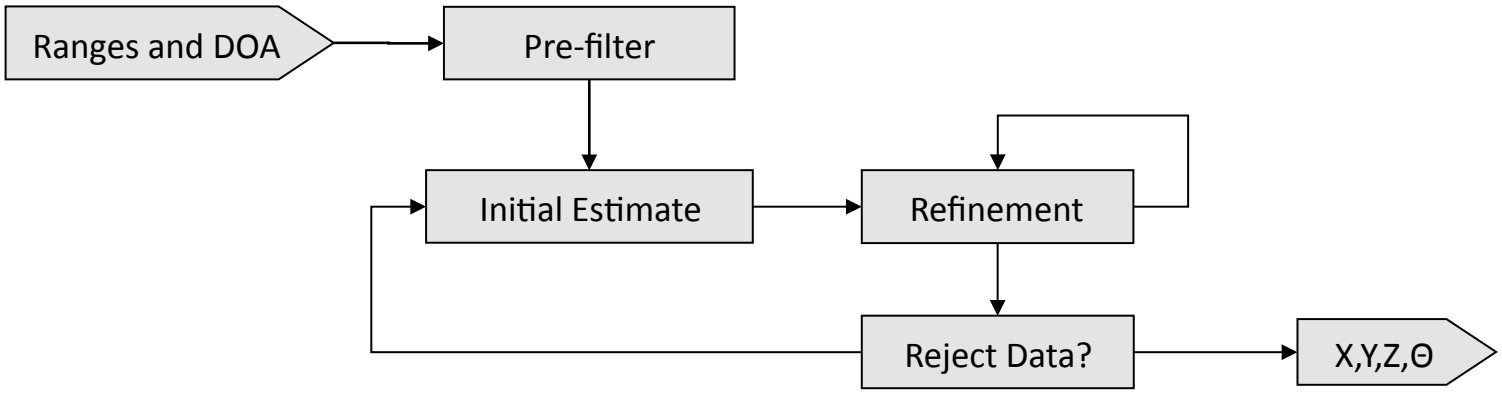
Position Estimation

- Problem:
  - Given pair-wise range and DOA estimates
  - Estimate X,Y,Z locations and orientation Θ for each node

R,θ,ϕ

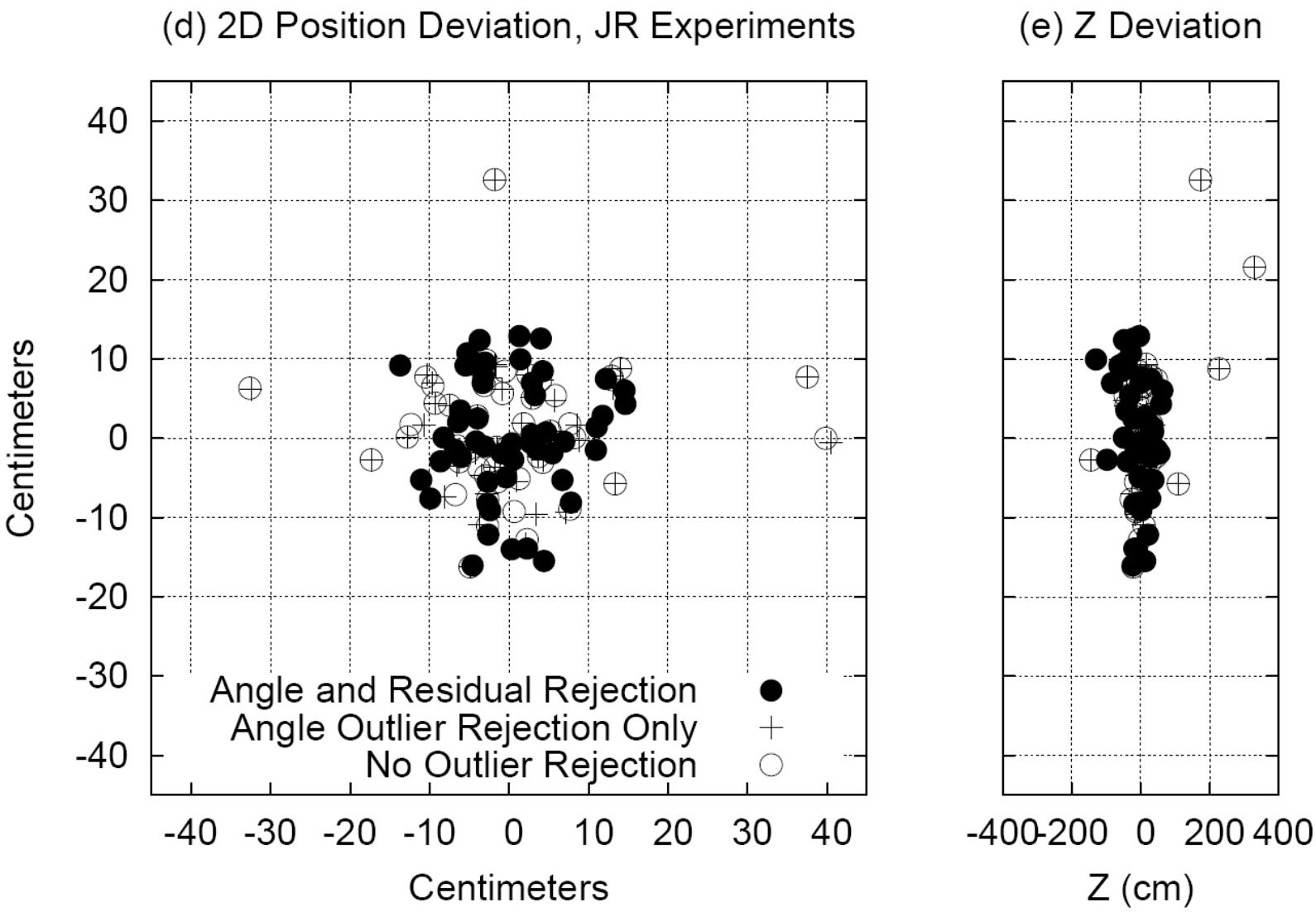


Position Estimation Solution



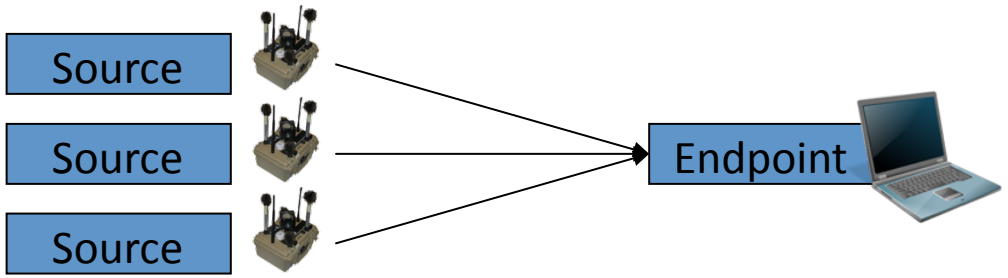
- Iterative refinement
  - Non-linear least squares position estimation
  - Interleaved orientation estimation
- Filtering out bad data is key to good results
  - Pre-filtering step
  - Over-constrained system → reject inconsistent data

Self localization accuracy



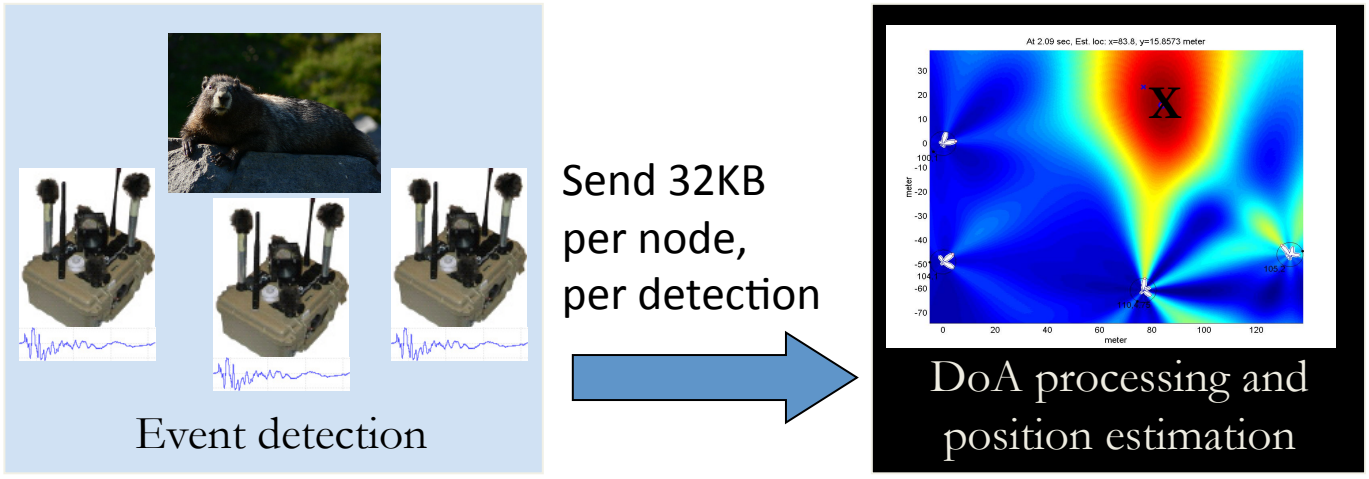
VoxNet Applications

- Programming language: Wavescript
  - High level, stream-oriented macroprogramming language
  - Compiles down to a directed graph of communicating 'stream operators' which data flow through (source to endpoint)
  - Operates on stored OR streaming data
- User writes a script connecting operators
  - Writes both node side + sink side separately
- User decides where processing occurs (node, sink)
  - Explicit, not automated processing partitioning



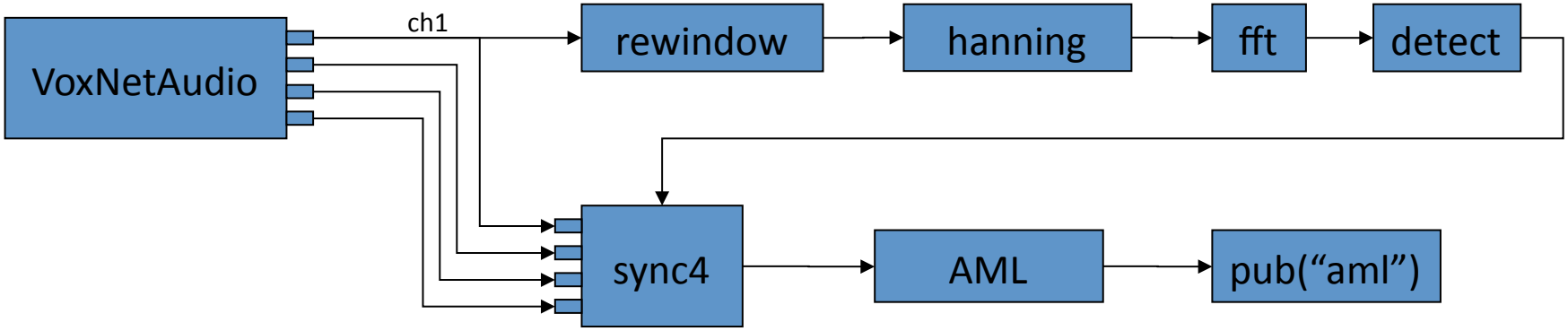
Example VoxNet application

- Detect and determine location of marmots
  - Combine DoA estimates from surrounding nodes
  - Obtain results on-line



NODES SINK

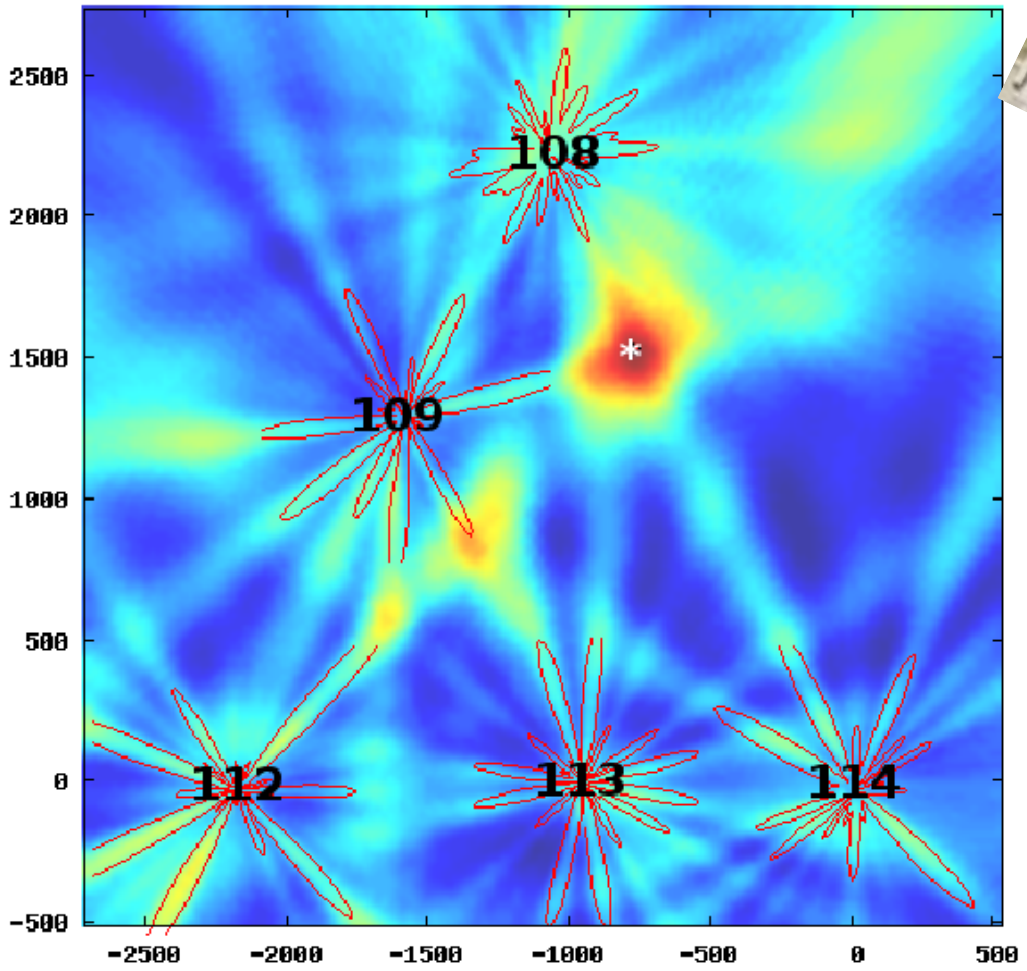
Demo Application: Streaming data plus AML direction finding



VoxNet Node

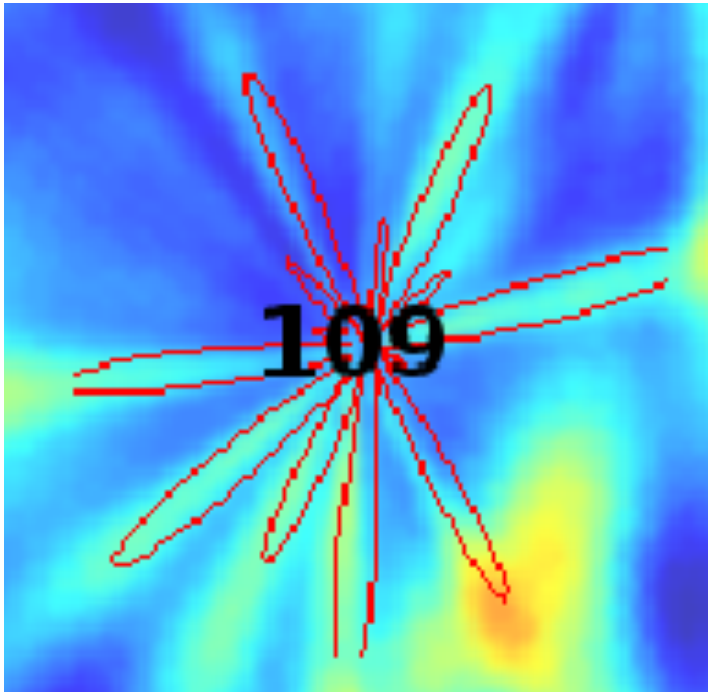
Laptop

IPSN Demo Video

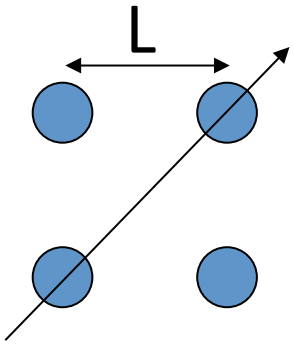


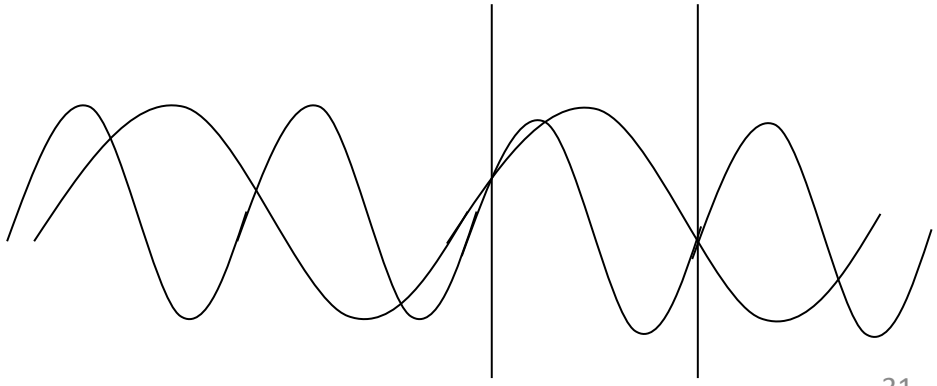
- Outdoor demo, behind Media Lab
- System deployed and auto-calibrated
- Human in field blows dog whistle
- System detects whiste, localizes source in real time

Spatial Aliasing



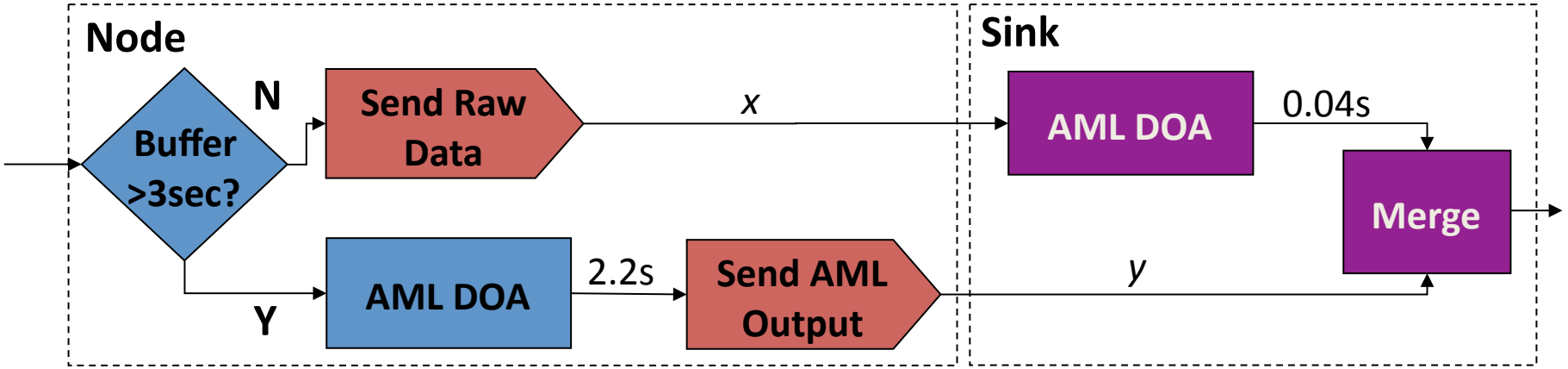
- Ambiguities in direction can occur if signal $\lambda < L/2$
- Spurious "Grating lobes"
- Filter using multiple nodes





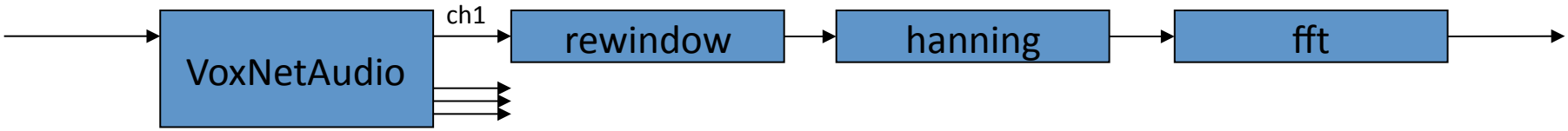
Applying the trade-off at run-time

- Aim: process raw data locally when possible
  - Node has limited buffer space for data samples
  - Event detection can happen faster than real-time, so can catch up



- Wavescript makes this easy to implement
  - Trivial, natural to split and merge multiple streams
  - Same on-node and sink DoA implementations

Example – energy calculation



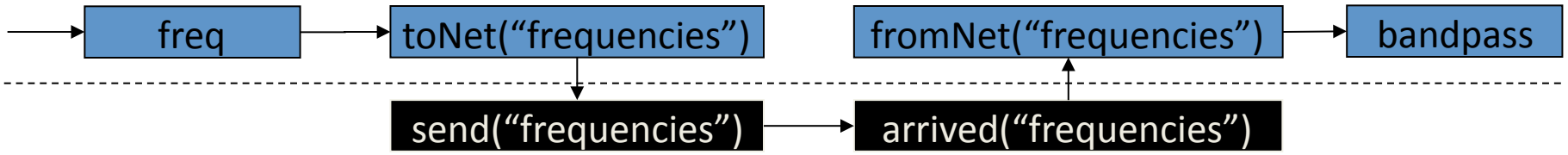

```
// acquire data from source, assign to four streams
(ch1, ch2 ch3, ch4) = VoxNetAudio(44100)

// window ch1 data stream into 32 sample chunks,
// passing through hanning then fft operators
freq = fft(hanning(rewindow(ch1, 32)))

// apply bandpass filter to freq
bpfiltered = bandpass(freq, 2500, 4500)

// calculate energy in bandpassed data
energy = calcEnergy(bpfiltered)
```

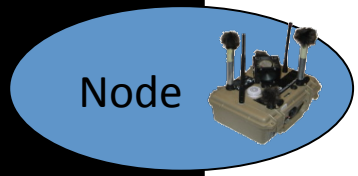
Splitting up a logical program




```
// acquire data from source, assign to four streams
(ch1, ch2 ch3, ch4) = VoxNetAudio(44100)

freq = fft(hanning(rewindow(ch1, 32)))

//send data to sink
toNet("frequencies",freq)
```

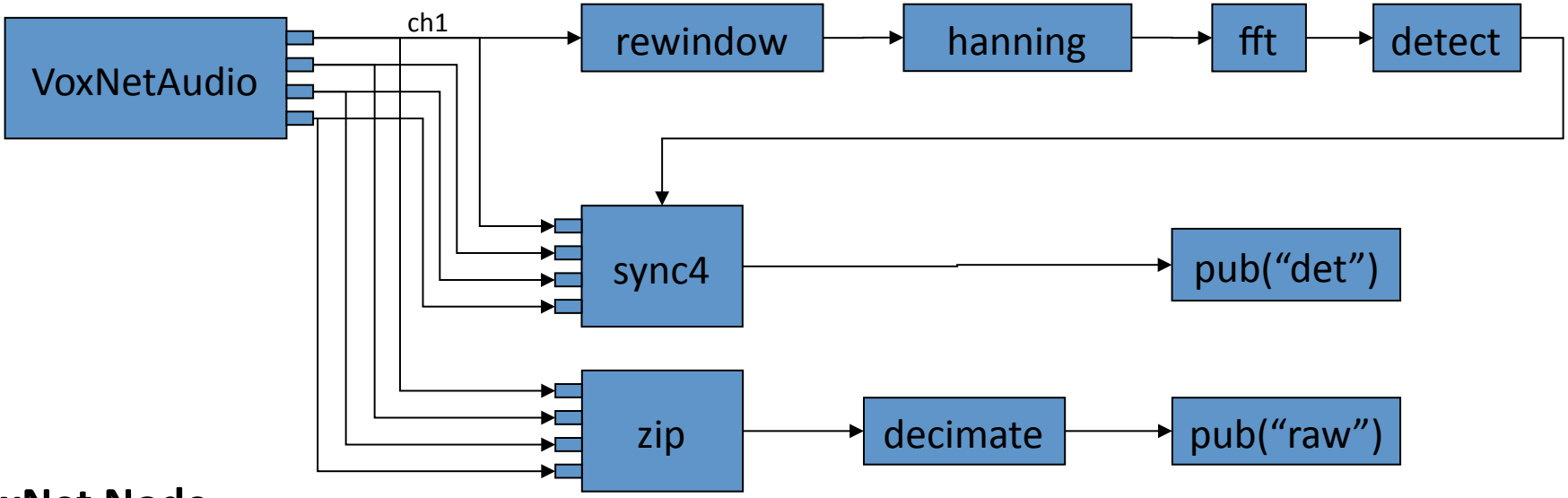



```
// receive data from node stream
freq2 = fromNet("frequencies")

// apply bandpass filter to freq
bpfiltered = bandpass(freq2, 2500, 4500)

// calculate energy in bandpassed data
energy = calcEnergy(bpfiltered)
```

Demo Application: Streaming data plus AML direction finding



VoxNet Node

Laptop