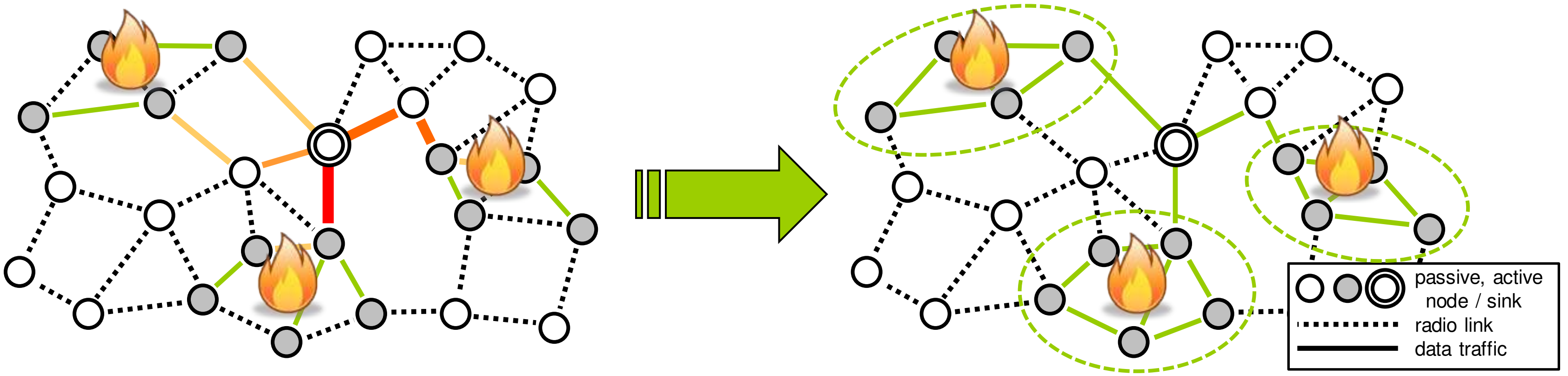
A System for Distributed Event Detection in Wireless Sensor Networks

Georg Wittenburg, Norman Dziengel, Christian Wartenburger, and Jochen Schiller
Freie Universität Berlin

9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN '10)
Stockholm, Sweden

Introduction / Motivation

- Key feature of WSNs: In-network data processing
  - ➢ Reduce communication between nodes and base station
  - ➢ Extend network lifetime



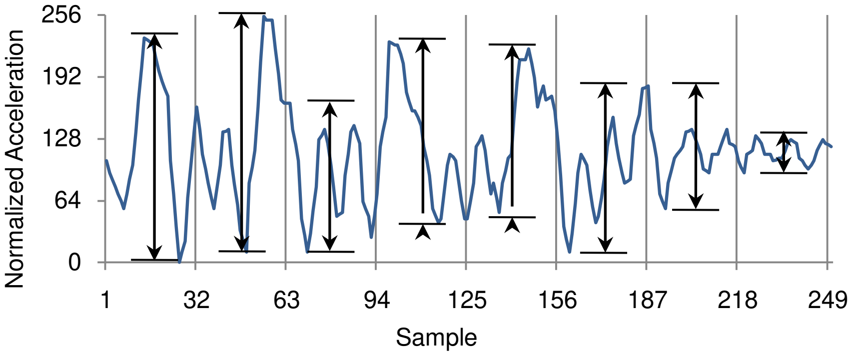
- One alternative: General-purpose event detection
  - Decide locally whether an application-specific event occurred (e.g., “There’s a fire!” or “A patient stumbled and fell!”)
  - Only transmit confirmed events to the base station
    - Avoid sending raw data from sensors

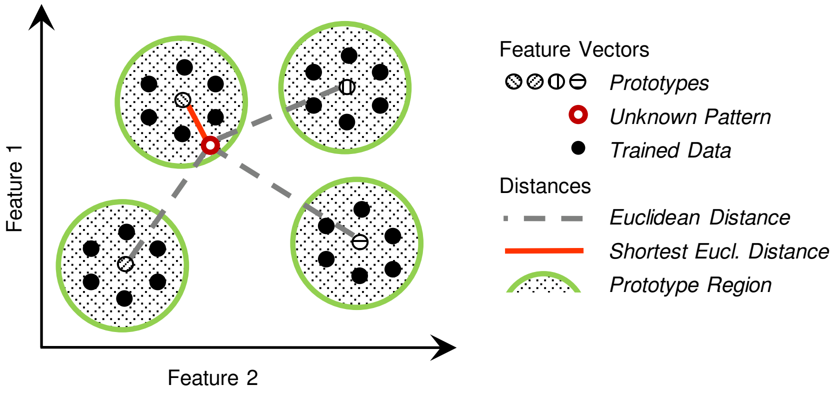
Use Case: Fence Monitoring

Freie Universität Berlin

- Sensor nodes attached to fence measure acceleration to detect security-relevant events (e.g., intruder climbing over fence)
- Realistic use case: Access control, perimeter security, ...
- Suitable properties:
  - Non-scientific users, i.e., not interested in raw data
  - No mobility, i.e., meaningful node positions
  - Potentially large deployments, i.e., long routes to base station

Basic Approach: Pattern Matching

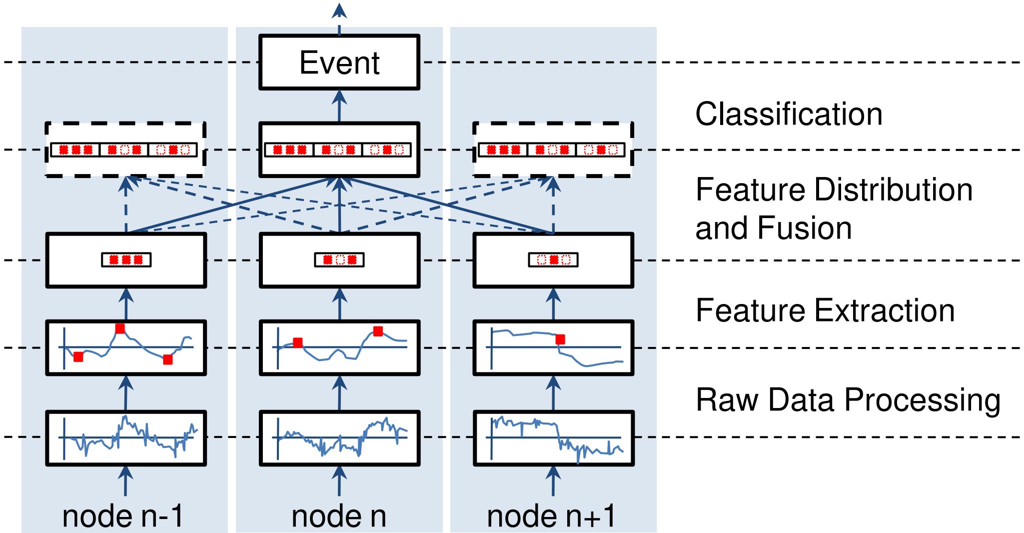




1. Feature Extraction:
   - Extract set of descriptive features from sampled raw data
   - Examples: Minimum, maximum, average, amplitude, duration, histogram, Fourier transform, ...
- Example:
   - Amplitude values extracted from acceleration data
   - ➢ Good properties:
      - Very descriptive in light of type of sensor and use case
      - Can be extracted without storing raw data

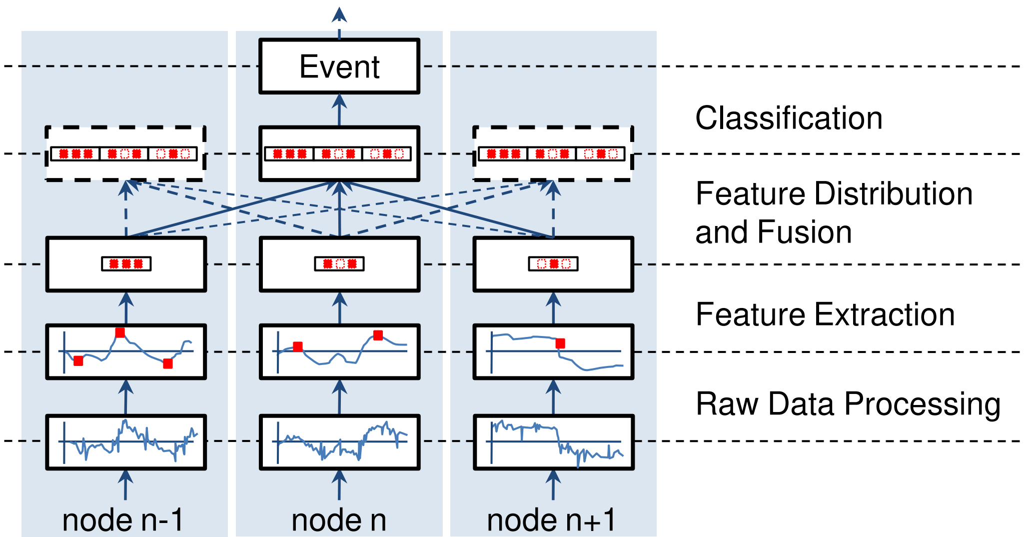
2. Classification:
   - Use extracted features to deduce previously trained event
   - Combine features into *feature vector* and compare to *prototype vectors* of events
- Example:
   - Four prototype vectors established by averaging training data
   - Classify feature vector by finding nearest prototype vector
      - If feature vector is close enough to prototype, event is recognized
      - Distance to prototype indicates confidence of classification

Event Detection in WSNs (1/2)



1. Raw Data Processing:
   - Periodically sample sensors
   - Filter, normalize, and smoothen data
   - Control sampling frequency
     - Preserve energy in phases of inactivity
2. Feature Extraction:
   - Extract application-specific set of features from raw data
   - Selection of appropriate features is part of training

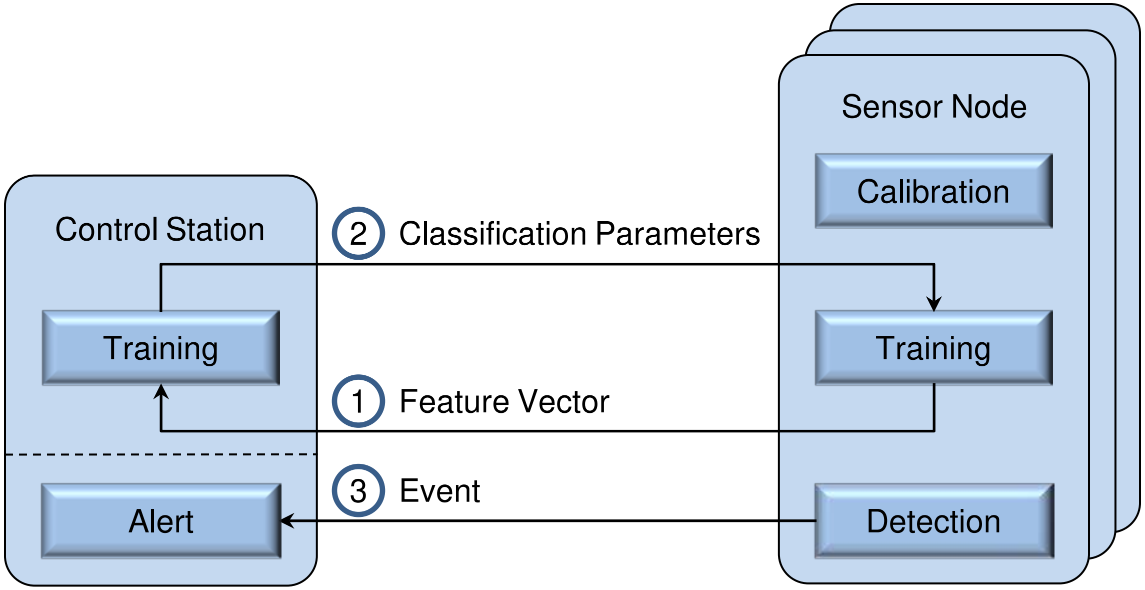
Event Detection in WSNs (2/2)



3. Feature Distribution / Fusion:
   - Broadcast features to *n*-hop neighborhood
     - Usually *n* = 1 because radio range exceeds expansion of events
   - Retransmit features in case of transmission failures
     - Nodes may fail to receive packets during feature extraction due to processing load

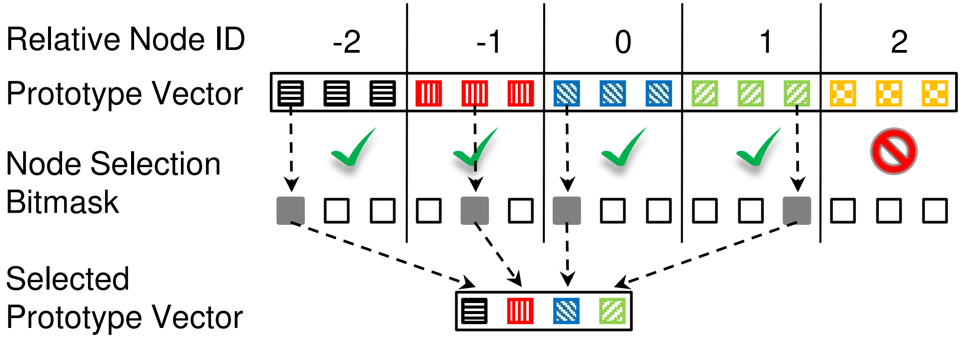
4. Classification / Reporting:
   - Combine local and received features into feature vector
   - Classify feature vector
   - If event is configured as relevant, report it to base station
     - Otherwise, locally log event for user-initiated retrieval
   - Base station fuses classification reports if necessary

System Overview: Training / Detection



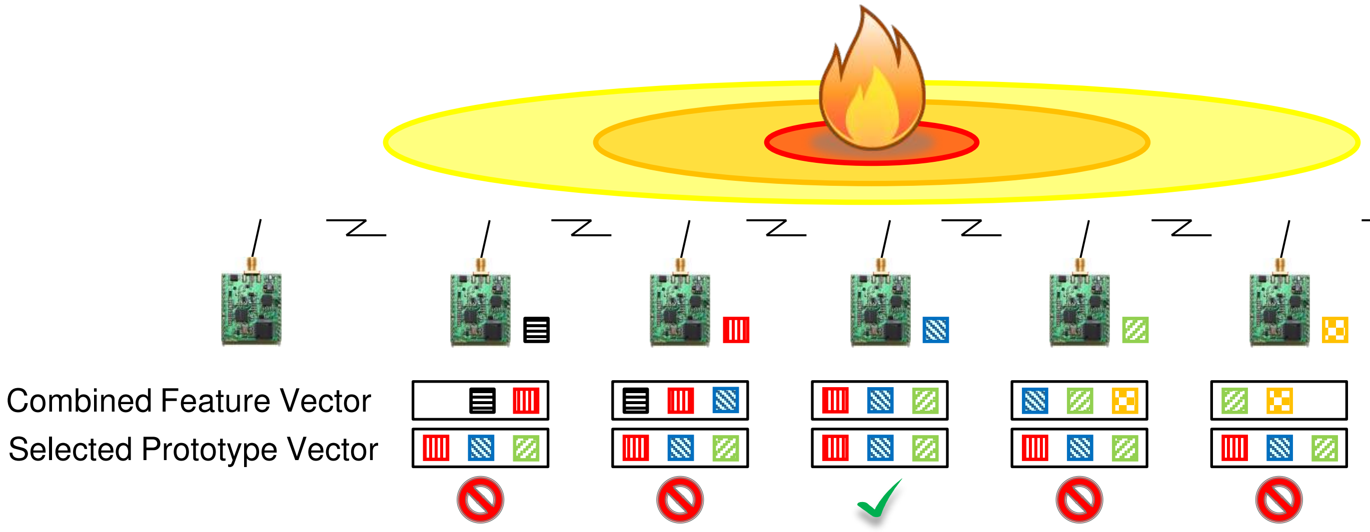
1. Training
   - Expose sensor network to series of training events
   - Extract all supported features and transmit them to control station
2. Setup
   - Select best subset of features, calculate prototype vector for each event
   - Configure nodes to only extract/transmit selected features, setup prototype vectors
3. Event Detection
   - Detect and report events

Feature Selection (Overview)



- Advantages of reducing the number of features:
  - Less computation required on nodes
  - Less data needs to be transmitted
    - ➢ Saves energy, reduces probability of packet loss
- Two selection steps:
  - Only consider features that are detected reliably
    - ➢ Ensure that physical effects of event are pronounced enough at given distance from center of event
  - Select only high quality features, i.e., those that result in distinctive prototype vectors

Brief Example



- Setup:
  - Nodes in a line, one feature extracted per node
- Nodes are configured to recognize one single event
  - Identified by prototype vector with three features:
    1. Feature from neighboring node on the left
    2. Feature from local node
    3. Feature from neighboring node on the right
- Event detection (on all nodes):
  1. Sample and process raw data
  2. Extract feature(s),
  3. Distribute features and calculate feature vector
  4. Perform classification
- Feature vector only matches prototype vector on node at location of event
- Central node detects (and reports) event; other nodes ignore event

Experimental Evaluation – Setup

- Sensor nodes attached to fence of construction site
  - One node per fence element (3.5m wide, 2m high)
- ScatterWeb MSB sensor node:
  - TI MSP430 16-bit microcontroller (5 KB RAM, 55 KB flash)
  - ChipCon 1020 radio transceiver (operating at 868 MHz)
  - Freescale Semiconductor MMA7260Q 3-axis accelerometer
- Four different events
  - Trained and evaluated with 15 samples per event

Experimental Evaluation – Events

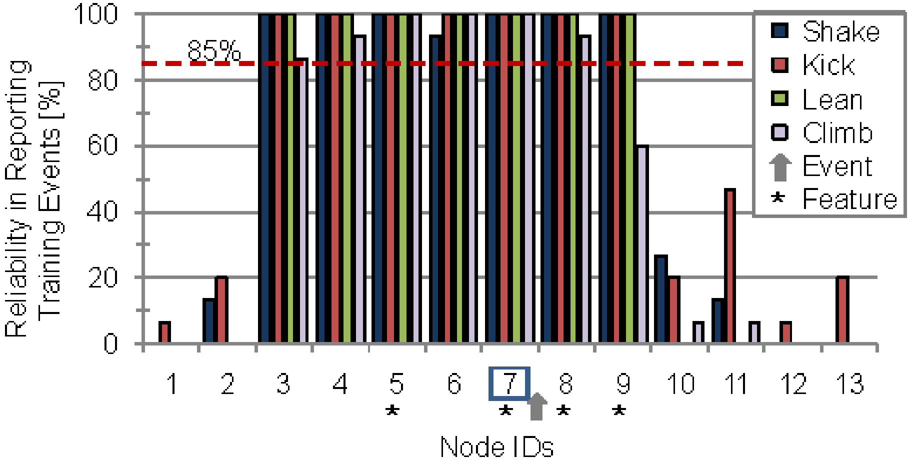
Shake

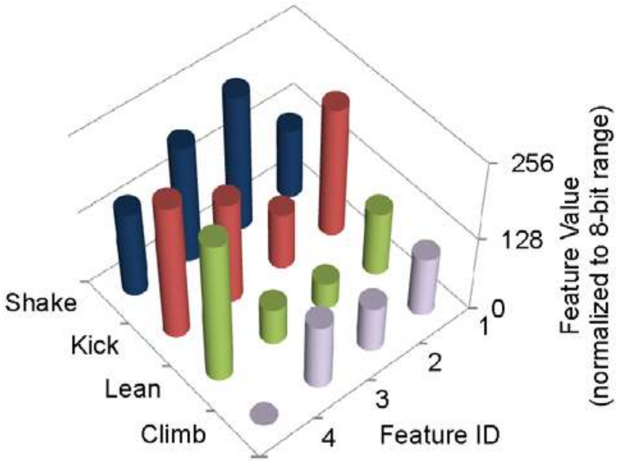
Kick

Lean

Climb

Results – Feature Selection





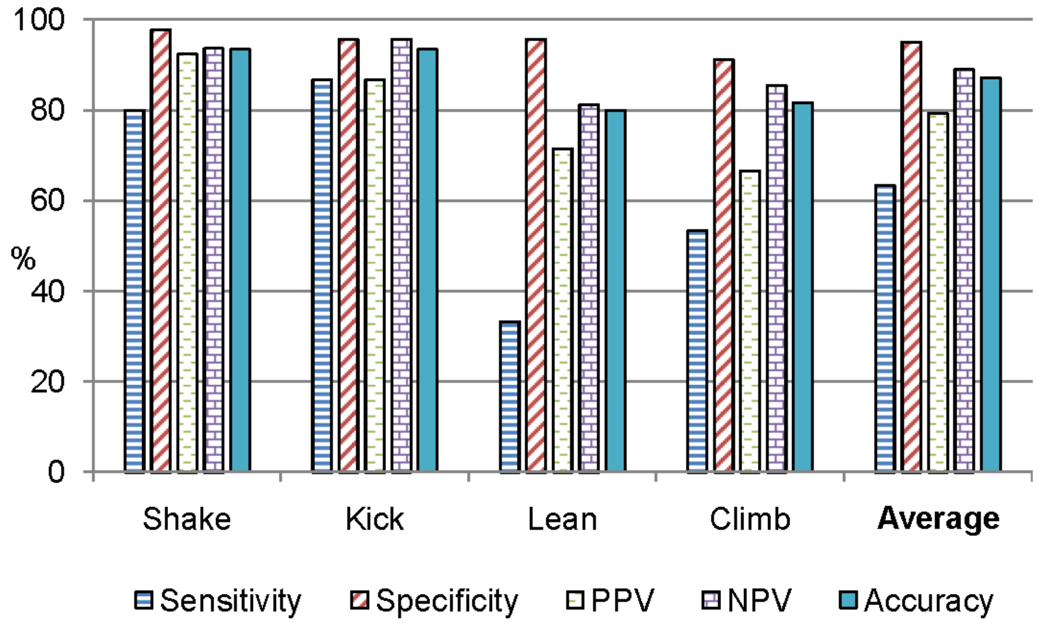
- Seven nodes were reproducibly affected by events
  - Reliability above threshold of 85%
- Features from nodes # 3 to # 9 are deemed reliable enough
- Quality-based feature selection results in four features

- Events do not propagate evenly in both directions on the fence

- Selected features:
  - ID # 1: Histogram feature from node # 5
  - IDs # 2 to # 4: Amplitude features from nodes # 7, # 8, and # 9
- Selected nodes are close to location of event
- Each prototype vector differs from any other one in at least one feature

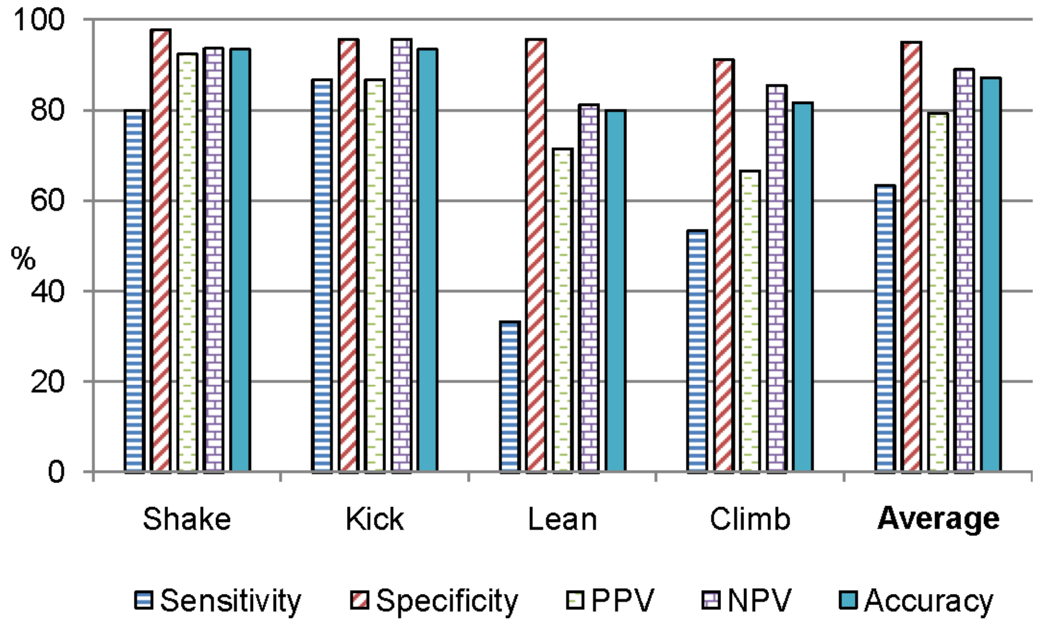
- Feature selection compensates for unevenness in propagation characteristics

Experimental Evaluation – Metrics



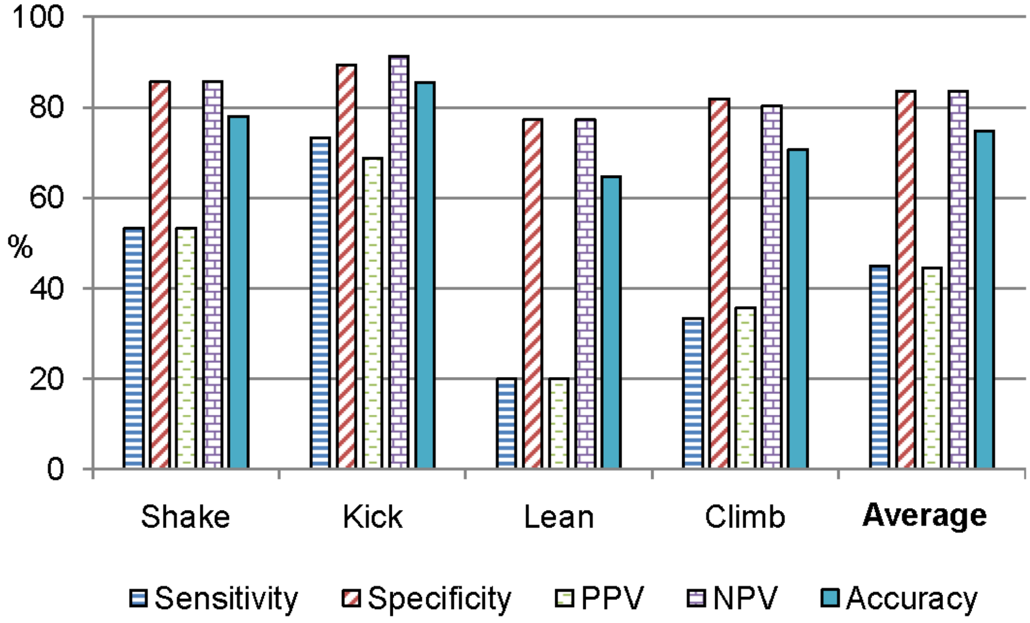
- Sensitivity (recall) = TP / (TP+ FN)
  - Proportion of correctly detected events in all events of that type
- Specificity = TN / (TN+ FP)
  - Proportion of correctly ignored events in all events of another type
- Positive Predictive Value (PPV, precision) = TP / (TP+ FP)
  - Proportion of correctly detected events in all detections of that type
- Negative Predictive Value (NPV) = TN / (TN+ FN)
  - Proportion of correctly ignored events in all detections of another type
- Accuracy = (TP+ TN) / (TP+ TN+ FP+ FN)
  - Proportion of true results in the population

Results – Feature Fusion



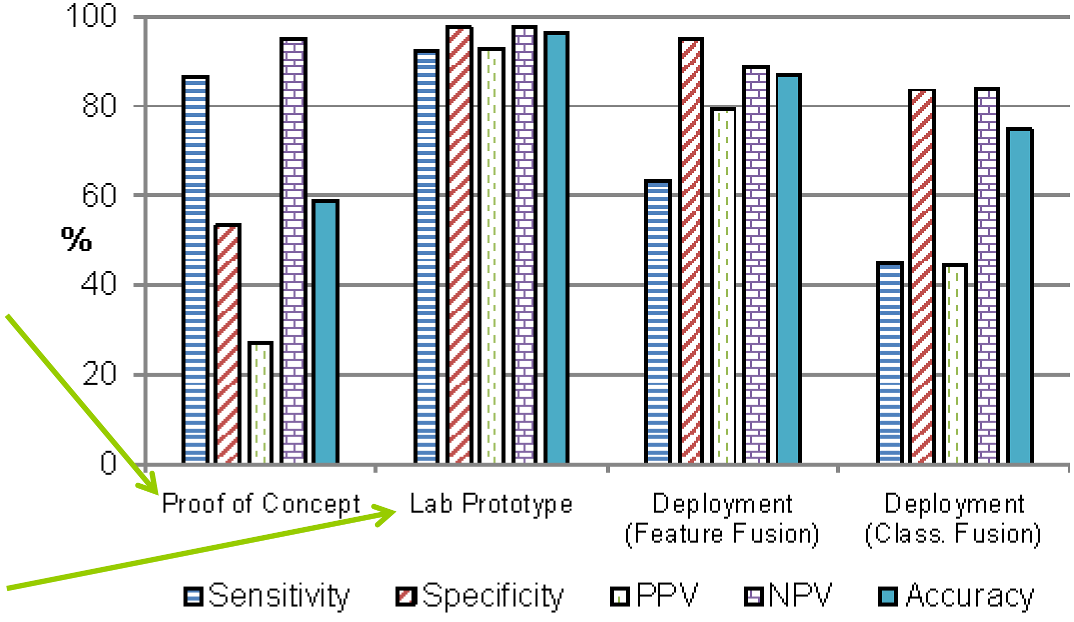
- Shake and kick events detected reliably
  - All metrics above 80%, accuracies of 93.3%
- Detection of lean or climb events not as accurate
  - Sensitivity is comparatively low, while specificity remains high
  - Too many events are falsely rejected due to prototype regions being too small
  - Training runs were too similar to each other, prototype regions only enclose part of required space
- Overall accuracy of 87.1% after feature fusion

Results – Classification Fusion



- Specificity, NPV, and accuracy decrease slightly; sensitivity and PPV decrease considerably
- Base station counts incorrect classification from other nodes, if
  a) correct classification is falsely rejected on central node, while incorrect classification is reported from another node
  b) node reports incorrect classification higher confidence than that of correct classification
- Overall accuracy of 74.8%

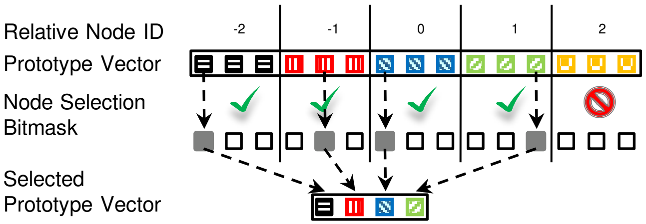
Comparison with Prior Work

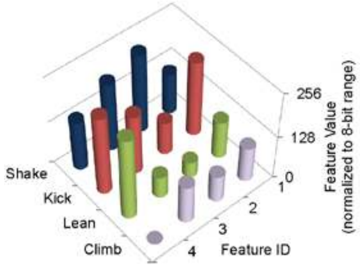


- Improvement over proof-of-concept implementation
  - Rule-based classifier, accuracy of 58.8%
  - Improvement of 28.8% (feature fusion, classification fusion was not supported)
- Unable to reach same level of accuracy as lab experiments
  - Manual feature section, accuracy of 96.3%

Conclusion

- System for distributed event detection in WSNs
  - No external coordination or processing required
  - Trainable to detect different classes of application-specific events





- Event detection accuracy shows improvements over prior work
  - Setup of experiments leaves room for further improvement
- Open questions:
  - Energy efficiency: Purpose-built sensing platform under development
  - Applicability: Medical applications, complex surveillance, ...

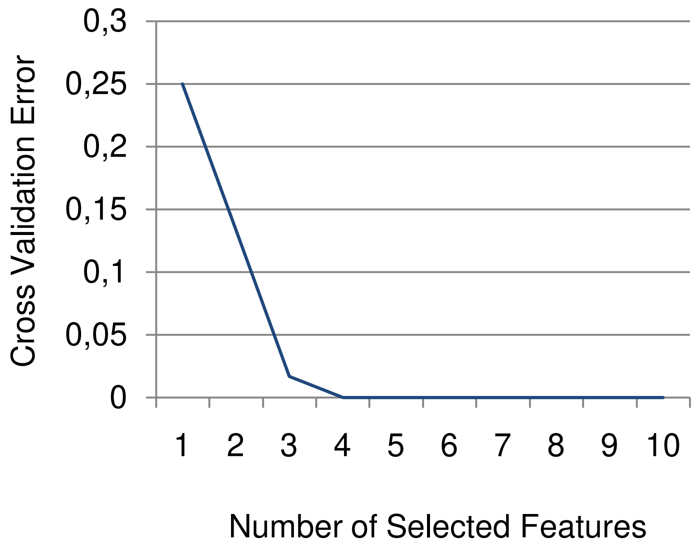
Outline

- Event Detection in WSNs
  - Distributed Pattern Matching
  - Feature Selection
- Brief Example
- Deployment / Evaluation

Feature Selection (Feature Quality)

- Leave-one-out Cross Validation (LOOCV):
  1. Iteratively pick one training feature vector from set of vectors
  2. Calculate prototype vectors using remaining vectors
  3. Check the classification error of prototype vectors using selected vector
  4. Iterate over all possible vectors to pick, average classification errors

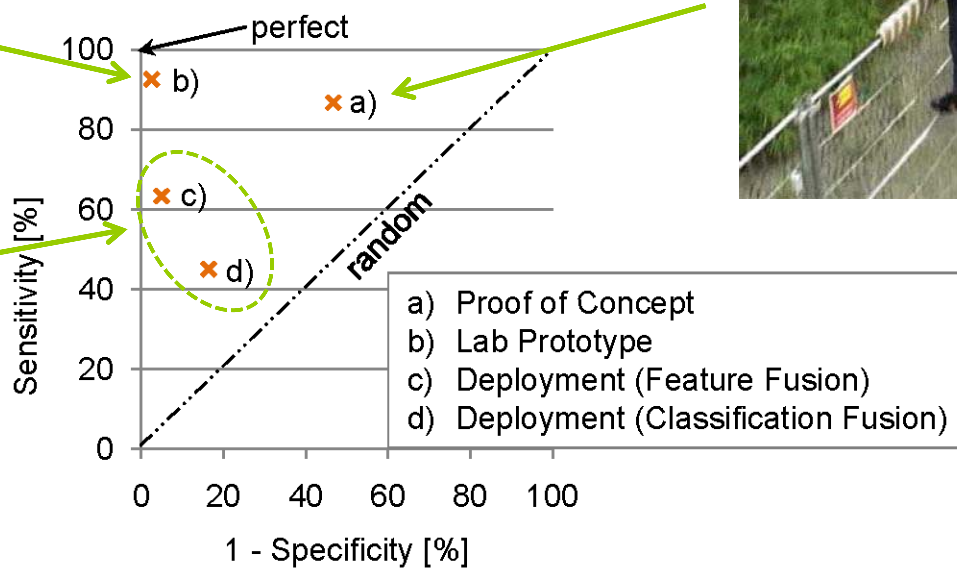
➢ LOOCV averaged classification error serves as quality metric for features



- Feature selection algorithm:
  1. Start with empty set of features
  2. Greedily select feature with largest reduction in LLOCV error
  3. Add this feature to set of selected features
  4. Repeat until no additional feature results in noteworthy reduction of error

➢ Configure sensor nodes with resulting set of features

Comparison with Prior Work



Sources of Error

Damping

Disconnect

Sensor Orientation

Fence Configuration

Lessons Learned

Problem #1: Non-uniform setup

- Irregularities in fence setup fence as deployed by construction workers
- Physical effects of events do not propagate evenly in all parts of deployment area
- Violates fundamental assumption

Solutions:

- Only deploy system in scenarios with uniform propagation characteristics
  - ➢ Take greater care to properly connect fence elements to each other
  - ➢ Unpractical for production-level system, may require additional training of workers
- Train the events on several locations of the deployed system
  - ➢ Calculate prototype vectors based on data reported by sensor nodes in different parts of deployment area

Problem #2: Familiarity with events

- Events were trained in strict order
  - (15 x shake, 15 x kick, 15 x lean, 15 x climb)
- Test subjects became familiar with setup as training progressed
- Sample events grew similar to each other, size of prototype regions decreased
- ➢ Lower sensitivity for lean and climb events

Solutions:

- Increase numbers of test subjects and/or sample events
  - ➢ Training requires even more time
- Change training process to train one sample event of each class
  - ➢ Avoid bias in size of prototype regions without committing additional resources