Leveraging Localisation Techniques for In-Network Duplicate Event Data Detection and Filtering

Jakob Pfender and Winston K.G. Seah
School of Engineering and Computer Science
Victoria University of Wellington, Wellington 6140, New Zealand
Email: { jakob.pfender, winston.seah }@ecs.vuw.ac.nz

***Abstract*—Event detection has become a ubiquitous application in the domain of wireless sensor networks. In any distributed event detection system, duplicate event data that can increase the likelihood of network congestion is a primary concern, and in-network data aggregation is a popular approach to alleviate this problem; similarly, duplicate data can be discarded. We propose a novel approach that utilises localisation techniques to quickly identify and discard duplicates among event reports. Using just the locations of nodes reporting events, the proposed classifier is able to make a decision on which event reports are to be discarded as duplicates. We show that our approach has the potential to greatly reduce packet load in a network and thus save energy and increase sensor life.**

I. Introduction

Consider a simple wireless sensor network (WSN) used for distributed event detection. Assume that we want to use the network to quickly determine the general locations of ubiquitous and frequently occurring events, such as hotspots in an area prone to forest fires. Detailed information about detected events, such as precise sensor readings or the size of the affected area, is irrelevant; we simply want to know where and when events have occurred and to be able to distinguish between them. However, if events happen to occur in close proximity to one another at the same time, the network does not necessarily need to distinguish between them, since it is sufficient to know that an event has been reported in this general area at this point in time, even if that event may in reality be two or more closely neighbouring events.

Upon sensing an event, a node sends an event notification to a fixed sink node using a given routing algorithm. The event notification contains only the location of the sensing node and time of occurrence as recorded by the sensing node. Other information about the event may be sent but is only relevant to the application at the sink or beyond. Assuming that events are ubiquitous and are typically detected by several nodes at roughly the same time, it is reasonable to assume that at any given point in time, there is a large number of event notifications en route within the network. However, since many of these notifications concern the same events, the network is more congested than it needs to be, since we only require one notification for each distinct event to be received at the sink.

Thus, the challenge is to quickly and reliably identify which of these event notifications reference the same event and can therefore be discarded. We propose a novel way of classifying event notifications using only the sensing nodes' locations as inputs. The classifier proposed in this paper is primarily based on the concepts of *trilateration* and *multilateration*, i.e. estimating a location by using distance measurements to three or more known locations. Trilateration derives a location estimate by finding the intersection of three circles whose centres correspond to the known locations and whose radii correspond to the measured distance to these locations. Fig. 1 shows an example scenario. If a node with unknown position has distance measurements from nodes a, b, and c, whose positions are known to us, we can draw circles corresponding to the measured distances around their locations and compute the intersection between those circles. The measuring node's position will then be somewhere in the grey area. More advanced methods using more than three known locations in order to reduce the size of the intersection and generally produce more reliable results are known as multilateration; these are mainly based on maximum likelihood estimation and regression analysis methods [6], [13], [16].

Considerable amounts of research have been conducted in order to refine and improve upon the simple trilateration in

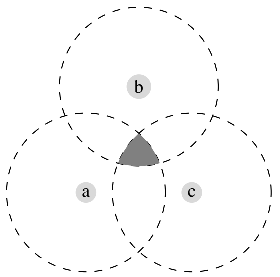


Fig. 1. Example of location estimation by trilateration

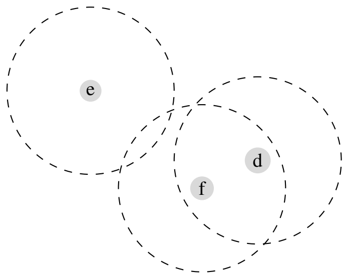


Fig. 2. An unsuccessful trilateration attempt

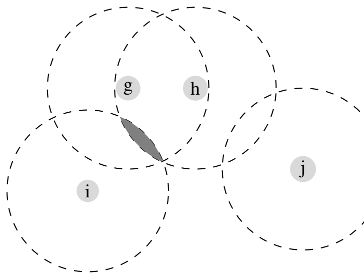


Fig. 3. Removing outliers (e.g. j) can improve results

hopes of achieving better and more accurate location estimates [5]. As we will show in this paper, it is possible to apply the basic aspects of multilateration to event reports in order to identify duplicate reports.

The rest of this paper is organised as follows. We first describe the basic concept behind our classifier in Section II-A. We then show that our proposed approach is sound and that it achieves good accuracy and precision in its classification results (Section III). Following which, we evaluate our approach by comparing it to other distributed event detection techniques both in terms of complexity of the approach and quality of the results (Section IV). Lastly, we discuss the benefits and drawbacks of the proposed approach, and provide suggestions on further refinement and improvement.

II. The Classifier

A. Basic idea

Our basic assumption regarding event notifications is that only the location of a sensing node is known. Using only this information and our knowledge of localisation techniques, we can design a simple classifier capable of making predictions as to whether a list of event notifications all refer to the same event. Note that this is not classification in the strict mathematical sense because we are not directly sorting observations into *a priori* categories; however, in Section III we evaluate the performance of our solution using methods commonly used to validate classifiers, which is why we use the term here.

To design the classifier, we make use of the fact that our sensing nodes have a known *sensing distance*, i.e. a maximum range within which they can sense events. If at any point in the network, a node holds three or more event notifications from different nodes in its packet buffer, it can simply perform a tri- or multilateration using the location information given in the notifications along with the known sensing range as inputs. This will give an estimate of the event location. This estimate can then be checked against the sensing range of each node that sent a notification.

If the location estimate is within sensing range of all reporting nodes (i.e. there exists an intersection between the sensing ranges of all nodes), as in the scenario shown in Fig. 1, we can assume that the event reports *probably* refer to the same event, in which case we can simply forward one of the reports as a representative and delete the rest as duplicates. Section II-B will discuss in more detail whether this assumption is justified.

On the other hand, if the estimated event location is outside the sensing range of one or more of the nodes, this means that the nodes cannot all be referring to the same event, since it is impossible (assuming the location estimate is sufficiently accurate) that they are all in range to sense it. Fig. 2 shows such a scenario: If nodes d, e, and f all register an event at roughly the same time, it is unlikely that they are referring to the same event, since the trilateration using their positions and sensing ranges would not be able to produce an intersection between all three circles. A sophisticated localisation algorithm might produce a location estimate somewhere between nodes e and f; however, this estimate would lie outside the sensing range of at least one of the nodes and would thus not be valid. Since it is impossible to tell which of the reporting nodes are referring to the same event given this information, we make no decision and simply forward all three event reports.

Algorithm 1 The classifier (called when attempting to clear the buffer)

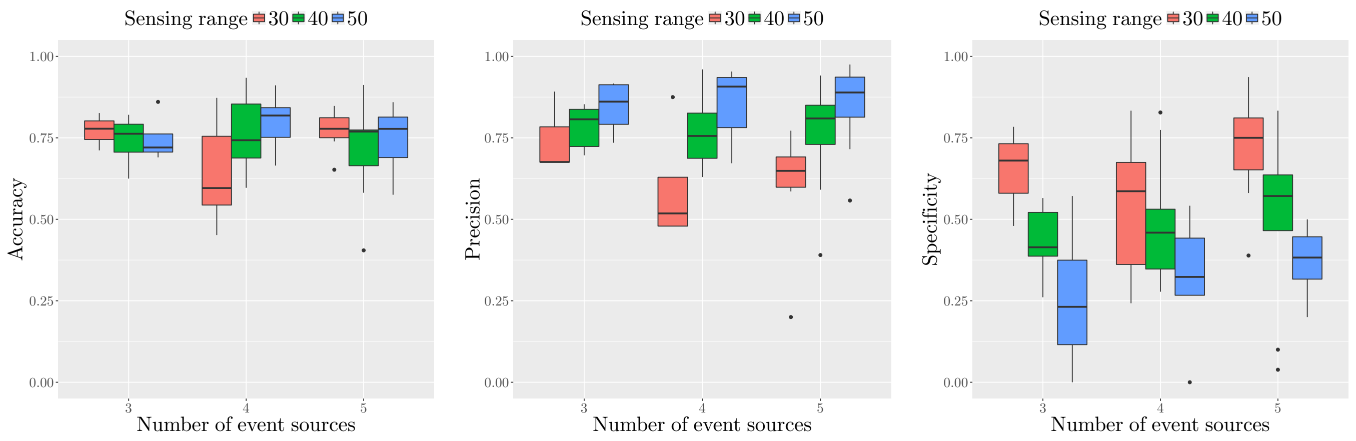
```
 1: function CLASSIFY(buffer)
 2:     while buffer.length $\geq 3$ do
 3:         estimate $\leftarrow$ multilateration(buffer)
 4:         if is_valid(estimate) then
 5:             forward(buffer.front)
 6:             buffer $\leftarrow \emptyset$
 7:         else
 8:             identify furthest outlier
 9:             forward(outlier)
10:             buffer $\leftarrow$ buffer $\setminus$ outlier
11:         end if
12:     end while
13:     for all packet $\in$ buffer do
14:         forward(packet)
15:     end for
16:     buffer $\leftarrow \emptyset$
17: end function
```

If we have event reports from more than three nodes, there is an additional step we can take to try and filter some of the reports: If the initial multilateration attempt is unsuccessful (i.e. the reports are not all referencing the same event), we can identify the furthest outlier among reporting nodes, i.e. the node whose position is furthest from the others. We forward that node's report and then repeat the multilateration with the remaining nodes. This is beneficial in scenarios such as the one shown in Fig. 3, where report j is a clear outlier referring to a different event than the rest of the nodes. If we forward j's event report, the subsequent trilateration on g, h, and i will succeed and we can treat them as duplicates.

B. Classifier Details

Whenever a node senses an event, it dispatches an event notification packet containing its own location to the network. Depending on the underlying routing algorithm, this packet might be duplicated multiple times on its way to the sink by all nodes receiving it. This is where the classifier comes into play. Instead of blindly forwarding all received event reports, nodes will first store all received reports in a buffer. Nodes will then periodically attempt to clear their buffer using Algorithm 1.

If there are two or fewer reports waiting in the buffer, they will simply be sent out, since there is no way to discern whether they refer to the same event using trilateration and since two extra packets will not impact the network's congestion status in a significant manner. If there are three or more reports in the buffer, the classifier is invoked using all reports sent within a few seconds of one another as inputs. As described above, the classifier initially assumes all such



(a) Accuracy (b) Precision (c) Specificity

Fig. 4. Accuracy, precision, and specificity (an ideal perfect classifier would achieve 1.0 in all three aspects)

reports are referring to the same event and attempts to compute a location estimate for this event based on the reporting nodes' locations. This location estimate is then checked against the reporting nodes' locations and sensing ranges.

If the location estimate is not within sensing range of all node locations in the report buffer, we start removing outliers among the node locations one by one. We compute the mean of all report locations in the buffer and select the report whose location is furthest from this mean. We forward this event report and remove it from the buffer. We then repeat the invocation of the classifier on the rest of the buffer until the trilateration reports a valid result — i.e. all reports refer to the same event — or the buffer contains fewer than three reports.

Recall that we stated in Section II-A that if the locations of all reporting nodes are within sensing distance of the estimated event location, we assume that they are all referring to the same event. It is of course possible for two distinct events to occur at roughly the same time in close proximity to one another, so that both events are registered by the same set of nodes; this would lead to the classifier treating event reports generated by either event to refer to the same event. However, as we stated in Section I, it is not important for our scenario to distinguish between events occurring right next to one another at the same time, since one report on the fact that there was an event at a particular location is sufficient. Therefore, this apparent oversight by the classifier is within reason and helps to further reduce congestion by suppressing reports on closely neighbouring events that do not provide relevant additional information.

Since it is possible for event reports to be duplicated and sent on different paths on their way to the sink, the classifier might be invoked on a set of event reports that are identical (i.e., they were sent by the same node at the same time). Intuitively, one might consider introducing a preliminary check here in order to filter out such duplicates. However, we argue that this would not result in increased performance since it makes no difference whether the reports are compared before or during the classifier invocation; if we simply run the classifier on all reports as described above, it will filter identical reports implicitly.

III. Validation

We implemented our algorithm using Castalia [4], an open source WSN simulation framework for OMNeT++ [1].

Fig. 4 shows the classifier's accuracy, precision, and specificity [3], [10] when determining whether a given set of event reports are identical. We can see that the classifier is generally good at identifying duplicate event reports, but has a tendency to be overzealous and produce false positives. However, since our algorithm greatly reduces congestion by minimising the number of event reports sent through the network, we can achieve a positive trade-off between event reports dropped due to congestion and those dropped due to false classification.

We will show that this trade-off is beneficial in a future publication, along with more in-depth simulation results such as the effect of the classifier on overall network lifetime.

IV. Related work

The ideas discussed in this paper fall in the intersection of two research areas: distributed event detection and in-network data aggregation. We will give a brief overview of relevant contributions to these areas and place our work in context. In general, the approaches discussed in this section are more complex and less generalisable than our proposed approach, although they can achieve higher accuracy in their specific application domains.

A. Data aggregation

Fasolo *et al.* [9] provide a comprehensive overview of extant data aggregation techniques as well as an excellent taxonomy for classifying the different approaches taken within this research area. In the ten years since their survey was published, several promising new approaches have been developed, but their taxonomy is still well suited to catalogue them. In the terms used by Fasolo *et al.*, we can classify our approach as an *in-network aggregation with size reduction* that uses *periodic simple aggregation*, is *duplicate insensitive*, and relies on a *multi-path approach*. However, our approach simplifies some of the aspects Fasolo *et al*. consider necessary for data aggregation. In particular, we do not rely on data-centric routing, as the aggregation function is agnostic of the underlying routing mechanism, and we have no need for time synchronisation, as data is simply aggregated based on periodic buffer flushes at each individual node.

Most extant data aggregation protocols are *structured*, i.e. they rely on some form of in-network hierarchy, such as clusters or trees, in order to perform their aggregation. Among structured protocols, LEACH [11] is among the best-known. It is based on clustering where data are aggregated at specific points in the network known as *cluster-heads*, which then forward the aggregated data to the sink. This requires a setup phase before the actual event detection can begin, as well as some overhead during operation to ensure that the clusters stay consistent with the underlying topology.

In order to avoid the overhead inherent in structured algorithms, several unstructured approaches have been proposed. Among the first of these works was DAA+RW [8], which leverages MAC- and application-layer protocols in order to ensure spatio-temporal correlation of data and thus enables structure-free aggregation. Among structured approaches, the one closest to ours is Nath *et al*.'s Synopsis Diffusion [15], which is also a duplicate insensitive multi-path approach. However, the actual aggregation function is more complex than ours, requiring an overlay network in order to generate synopses.

B. Distributed event detection

The main focus of research in distributed event detection is on how a decision is made by a node or a set of nodes as to whether a set of measurements constitutes an event; however, there is significant overlap with data aggregation research, since most event detection schemes rely on some kind of data aggregation when making decisions and/or when forwarding event reports to the sink.

Krishnamachari and Iyengar [12] were among the first to propose a distributed solution to the event detection problem; they utilised Bayesian algorithms for event region detection. Luo *et al*. [14] expand on this approach while improving its effectiveness and efficiency.

Antonopoulos *et al*. [2] provide a comprehensive and up-to-date survey of current event detection approaches along with a taxonomy of techniques. According to them, most current approaches can be categorised as pattern matching, model based, or AI and machine learning based approaches.

Dziengel, Wittenburg, *et al*. [7], [17] propose a domain-specific pattern matching approach with tested deployments in fence surveillance and structural health monitoring. Their approach is highly effective at identifying events and fusing data, but the implementation is inherently application-specific, since it relies on extensive *a priori* classifier training using known event data.

V. Conclusions and future work

We have presented a novel approach to the problem of classifying event reports from a distributed event detection system by exploiting localisation techniques. The proposed classifier is much simpler than most extant data aggregation techniques and is effective at reducing packet load within a network; however, the quality of the classification results has room for further improvement.

With further refinements, it should be possible to achieve a level of quality in the results that, coupled with the energy savings and the fact that the approach is not application-specific, provides a strong incentive for using the classifier.

Future work will include an in-depth analysis of our simulation results in terms of classifier performance, network load, event delivery rate, and energy consumption.

References

[1] *OMNET++ Network Simulation Framework*. [Online]. Available: https://omnetpp.org/

[2] C. Antonopoulos, S.-M. Dima, and S. Koubias, "Event Identification in Wireless Sensor Networks," in *Components and Services for IoT Platforms*. Springer, 2017, pp. 187–210.

[3] A. Balazs, "International Vocabulary of Metrology-Basic and General Concepts and Associated Terms," *CHEMISTRY International*, 2008.

[4] A. Boulis *et al.*, "Castalia: A simulator for wireless sensor networks and body area networks," *NICTA: National ICT Australia*, 2011.

[5] T. J. Chowdhury, C. Elkin, V. Devabhaktuni, D. B. Rawat, and J. Oluoch, "Advances on localization techniques for wireless sensor networks: A survey," *Computer Networks*, vol. 110, pp. 284–305, 2016.

[6] J. E. Dennis Jr and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

[7] N. Dziengel, M. Seiffert, M. Ziegert, S. Adler, S. Pfeiffer, and J. Schiller, "Deployment and evaluation of a fully applicable distributed event detection system in Wireless Sensor Networks," *Ad Hoc Networks*, vol. 37, pp. 160–182, 2016.

[8] K.-W. Fan, S. Liu, and P. Sinha, "Structure-free data aggregation in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, 2007.

[9] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 14, no. 2, 2007.

[10] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006.

[11] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, Hawaii, USA, 2000, 10pp.

[12] B. Krishnamachari and S. Iyengar, "Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks," *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 241–250, 2004.

[13] K. Langendoen and N. Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison," *Computer Networks*, vol. 43, no. 4, pp. 499–518, 2003.

[14] X. Luo, M. Dong, and Y. Huang, "On distributed fault-tolerant detection in wireless sensor networks," *IEEE Transactions on Computers*, vol. 55, no. 1, pp. 58–70, 2006.

[15] S. Nath, P. B. Gibbons, S. Seshan, and Z. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 4, no. 2, p. 7, 2008.

[16] A. Savvides, H. Park, and M. B. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. ACM, 2002, pp. 112–121.

[17] G. Wittenburg, N. Dziengel, S. Adler, Z. Kasmi, M. Ziegert, and J. Schiller, "Cooperative event detection in wireless sensor networks," *IEEE Communications Magazine*, vol. 50, no. 12, 2012.