# Reliable localized event detection in a wireless distributed radio telescope

Suhail Yousaf*†, Rena Bakhshi*†, Maarten van Steen*†
*Department of Computer Science, VU University Amsterdam,
De Boelelaan 1081a, 1081HV Amsterdam, The Netherlands
s.yousaf@vu.nl, rbakhshi@few.vu.nl, steen@cs.vu.nl
†The Network Institute, The Netherlands

***Abstract*—We consider a large wireless network constituting a radio telescope. Each of the anticipated 3000 nodes is triggered to collect data for further analysis at a rate of more than 200 Hz, mostly caused by noisy environmental sources. However, relevant cosmic rays occur only a few times a day. As every trigger has an associated 12.5KB of data, and considering the size of the telescope in number of nodes and covered area, centralized processing is not an option.**

**We propose a fully decentralized event detection algorithm based on collaborative local data analysis, effectively filtering out only those triggers that need further (centralized) processing. As we show through performance evaluations, the crux in the design is finding the right balance between accuracy and efficient use of resources such as the communication bandwidth in the unreliable communication environment.**

***Index Terms*—distributed event detection, large-scale distributed systems, in-network data processing**

## I. Introduction

In a geospatial sensor network, a node is responsible for gathering location-sensitive data. In many cases, as these networks grow in diameter (measured in meters), as well as in their number of members, we often see that communication paths can be mainly realized only through wireless channels. A typical example that we consider in this paper is a large-scale radio telescope consisting of a few thousand nodes spread over an area of around 3000 $km^2$. An important consequence of this growth is that the processing of data needs to be localized, as realizing communication paths to centralized, specialized nodes becomes more difficult.

In this paper we consider a specific sensor network that we believe is characteristic for many other wireless sensor networks. First, each node is responsible for detecting specific, often rare, events, in our case detecting cosmic rays. Event detection requires sampling data. However, a node may be operating in an extremely noisy environment meaning that there can be many false detections leading to an explosion of sampled data of which most will turn out to be useless. It is crucial that such data is kept local and preferably not sent to any other node, let alone a centralized base station or the equivalent thereof.

Second, to distinguish between relevant and false events, a node requires information from its immediate (geographical) neighbors: an event is relevant only if some of the neighbors have detected it as well. This means that data from multiple nodes will need to be combined before coming to conclusions on relevancy of an event. In some sensor networks, the base station simply gathered the data from all nodes, but as we argued, this approach must be abandoned when networks grow.

We concentrate on the problem of distributed event detection in a large wireless sensor network (WSN) in which neighboring nodes collaborate to filter out relevant events before communicating associated data to one or several central nodes. Each node has not only a limited energy budget, but also other resources are scarce such as memory and processing capacity. In addition, as we are dealing with wireless networks; communication links are unreliable.

For our specific application, cosmic-ray detection, relevant events are so rare that we essentially cannot afford to lose any of them. In other words, much effort should be put into keeping the fraction of false negatives close to zero. On the other hand, false positives should also be minimized, but for a different reason, namely that of minimizing resource consumption and thus maximizing efficiency. However, optimizing for efficiency becomes more relevant when realizing that events occur at a rate of approximately 200 Hz, yet that less than 1% is actually relevant. In other words, a huge data-processing effort is required for successfully detecting cosmic rays.

We make the following contributions. First, we present a distributed, in-network event detection algorithm based on collaborative local data analysis that reduces resource consumption in large-scale geospatial sensor networks. Second, we investigate the application-level resource usage such as the communication bandwidth for a certain level of performance in the unreliable communication environment. This is the first paper to our knowledge to explore the possibility of applying collaborative local data analysis in large-scale geospatial wireless sensor networks to detect ultra-high energy cosmic rays.

The rest of this paper is organized as follows. In Sec. II, we give an overview of the related work in the field of the monitoring applications. Sec. III describes our distributed event detection algorithm designed for large-scale sensor networks. In Sec. IV, we evaluate the performance of our proposed algorithm in the presence of communication failures. Finally, Sec. V concludes the paper.

## II. Related Work

The common model for event detection in a WSN is that each node simply relays all of its locally generated data to the base station without local processing. The data is processed for event detection only at the base station [1]. This model works well for small-scale networks, a small amount of data per event, and lower frequencies of events per node. It is unacceptably inefficient for large-scale networks, as it may involve considerable bandwidth consumption as multiple communication hops need to be taken.

Another model for event detection involves in-network processing. To this end, processing is done by the nodes to compute events of interest against a criteria known to the node. This may significantly reduce the amount of communication and, hence, the energy consumed. The most commonly used techniques for in-network processing in WSNs are: (i) processing along a routing path to the base station [2]; (ii) processing at regional head nodes [3], [4]; (iii) initiating in-network consensus [5]; and (iv) deciding locally at a node based on information from its neighbours [6].

However, all the proposed schemes have different shortcomings for our application: (i) there is no acknowledgement of the event detection by a node in [2]; (ii) the scheme in [3] produces false negatives in case an event occurs on the border of two or more cells; (iii) a low event frequency is assumed in [6]; (iv) the schemes in [4] and [5] assume network-wide events, and thus, are not scalable for large geographical areas.

There are two classes of work done on cosmic-ray detection. A direct cosmic-ray detection method [7], [8], [9] needs a high-altitude balloon or a satellite/space mission, and detects only low-energy particles. To detect the much rarer highest energy particles, an indirect cosmic-ray detection method is needed, such as [10], [11], [12]. However, these systems use a wired backbone (fibre optic) for communication and, therefore, suffer seriously from geographical scalability issues.

## III. Distributed Event Detection

### A. Assumptions

In the context of cosmic-ray detection, we consider a large field covered by a large collection of stations, each equipped with a wireless sensor. They sense radio signals and communicate with neighboring stations in the field through a low-power wireless medium. Each station has limited processing capabilities, energy budget and a storage capacity in the order of a few hundred megabytes. The clocks of stations are globally synchronized via integrated GPS receivers. (The accuracy is within 1 to 2 nanoseconds through special devices [13].) Each station relays its data to a base station called the Central Radio Station (CRS) for further analysis.

The stations are stationary and location-aware. We assume direct communication only between stations within a certain distance (*geographical neighbors*). Each station captures radio signals with a certain strength into a so-called *L1 trigger*, which may indicate the occurrence of a cosmic-ray. Each trigger is timestamped at nanosecond accuracy. The timestamp is a pair of `seconds` (date and time into the UNIX epoch, in UTC) and `nanoseconds`. For each trigger, in addition to the timestamp, a digitized portion of the signal of 12.5 kilo bytes is also buffered at the station. This data along with the timestamp is sent to the CRS upon positive decision through a data analysis procedure; otherwise both the timestamp and buffered data are ignored.

The triggers of two geographically neighboring stations *coincide* if their timestamp difference $\Delta$T is less than $T_c$, the light-travel time in a straight line from one station to the other station. An L1 trigger is promoted to an *L2 trigger* if it is coincident with an L1 trigger of a geographical neighbor. An L1 trigger at a station is promoted to an *L3 trigger* in two cases: (1) the L1 trigger at a station is coincident with L1 triggers of at least two other geographical neighbors; (2) the L1 trigger at a station is coincident with an L3 trigger of any of its geographical neighbors. Note that we also call an L3 trigger an *event of interest*.

Ideally, if an L1 trigger at a station is not promoted to an L3 trigger, it is considered to be noise and must be discarded by the station. However, there is always a chance to discard an L1 or L2 trigger that is actually an L3 trigger, but due to communication failures could not be promoted to L3. This situation will result in *false negatives*. Moreover, a *false positive* is produced when a trigger that is not an L3 trigger is chosen to be reported to the CRS.

### B. The Algorithm

Whenever an L1 trigger occurs at a station, the station stores the L1 trigger locally and informs all of its neighbors by sending them the timestamp of its L1 trigger. Furthermore, when a station receives L1 triggers from its neighbors, it looks for a coincidence of the received triggers with its local ones. A station promotes its L1 trigger to an L3 trigger if its L1 trigger has coincidence with L1 triggers of at least two neighbors. To cover stations on the boundary of an event region with only one geographical neighbor in the event region, a station not only requires to broadcast its L1 triggers, but also to advertise its L3 triggers. This message helps a station promote its L1 trigger to an L3 trigger if its L1 trigger coincides with an L3 trigger contained in the advertisement message. To reduce bandwidth consumption, the algorithm uses periodic broadcast messages, which we call an *L1 bundle*, by simply grouping together the L1 triggers. Similar to the L1 bundle formation, the local L3 triggers are bundled as an *advertisement bundle*.

Figure 1 shows the pseudocode of our algorithm. When an L1 trigger occurs at a station $p$, it adds the trigger to its local cache. The trigger is also added to a local L1 bundle that will be broadcast to geographical neighbors of the station. The algorithm executes two threads: an active and a passive one. The active thread is executed periodically. It broadcasts L1 bundles and advertisement bundles of a station to the geographical neighbors of the station. The passive thread listens to incoming messages. Upon receipt of an L1 bundle or an advertisement bundle from a geographical neighbor, the thread looks for a coincidence of each trigger in the

```
/* On Local L1 Trigger */
localCache.add(L1(p))
L1Bundle.add(L1(p))

/* Active thread */
// Runs every T seconds
for all q ∈ Neigh_p do
   send < L1Bundle(p), q >
for all q ∈ Neigh_p do
   send < AdvertBundle(p), q >

/* Passive thread */
receive < L1Bundle(q), q >
OR
receive < advertBundle(q), q >

for all e ∈ Bundle do
   coincide = localCache.coincide(e)
   if coincide then
      L1(p) → L3(p)
      process(L3(p))
      if e ∈ L1Bundle then
         AdvertBundle.add(e)
      localCache.remove(L1(p))
```

Fig. 1: Pseudocode for our algorithm.

bundle with the local L1 triggers. Whenever an L1 trigger is promoted to an L3 trigger (see, $L1(p) \rightarrow L3(p)$ in Fig. 1), the station will execute the operation $process(L3(p))$ if further processing of the trigger is required ($e.g.$ applying any domain specific filter to the L3 trigger).

If a coincidence has been found between the local L1 trigger and L1 triggers of the geographical neighbors, then these L1 triggers are added to the advertisement bundle. On the other hand, if station $p$ promotes its L1 trigger to an L3 trigger because of coincidence with an L3 trigger of any of its geographical neighbors, the promoted L3 trigger is not broadcast to the neighbors. The reason for this is that station $p$ has only one neighbor in the event region which had already promoted its corresponding L1 trigger to L3. For station $p$, broadcasting the advertisement in this case is useless.

The local triggers are stored in a buffer with limited capacity. Depending on local processing capabilities and available memory, we need to take into account that the buffer may become full. As a consequence, a *cache eviction* strategy is required. For example, our algorithm can employ a simple strategy that discards newly arriving triggers while old triggers have not yet been removed. Alternatively, we could also decide to remove the oldest trigger, or choose one randomly. These strategies are rather straightforward, and are application oblivious. A more optimal cache eviction strategy may require further insight into the semantics of triggers. In this paper we take a simple approach and remove the oldest triggers when the buffer becomes full, except the ones that are currently being examined.

## IV. Performance Analysis

### A. Experimental Setup

To evaluate the performance of our event detection algorithm, we have conducted a set of experiments using the OMNET++ simulation environment [14]. In this subsection we explain the methodology used in the evaluation.

*1) Network Specification:* In the simulated network, the stations are set a few hundred meters apart from each other. To simulate the behaviour of a wireless ad-hoc network, we make a distinction between the so-called application layer and the system layer. The *application layer* consists of our basic algorithm with configuration parameters. The *system layer* is an application-independent wireless network. We are interested in obtaining insight in the resource requirements imposed by our application. In essence, our solution consists of two sets of algorithms. The first contains the core of our solution and its algorithms essentially assume that the system layer operates flawlessly and with infinite resources. The second set contains algorithms that compensate the shortcomings of the system layer: lossy links, faulty nodes, and limited resources. The core algorithms will always need to be executed; the compensation algorithms are executed only because the system layer is far from being perfect.

Executing the core algorithms will demand a certain capacity from the system layer in terms of bandwidth usage, memory usage, energy consumption, etc. Because of failures in the system layer, our compensation algorithms will also need to be executed, requiring further capacity. We are interested to know how much capacity the application layer requires from the system layer in the presence of faulty links and nodes.

To this end, we take a simple approach by initially assuming that only links can fail, in particular with a probability $p$. We furthermore consider only the use of bandwidth, as this is most likely our scarcest resource. We vary $p$ to see to what extent additional bandwidth is needed for the compensation algorithms. In this way, we aim at obtaining an upperbound for the bandwidth capacity that the system layer should provide. By refining the distribution of $p$, for example, by considering specific locations in the network, a more precise requirements specification can be obtained.

*2) L1 Triggers Traces:* We used traces of L1 triggers collected from a small-scale real testbed for cosmic-ray detection deployed by the Pierre Auger Observatory [11]. The testbed consists of 18 stations and uses a wired infrastructure for communication between stations and the central radio station (CRS). The data analysis procedure in the testbed is centralized: every station sends its L1 triggers to the CRS for noise filtering and further analysis. It is important to emphasize that the occurrence of L1 triggers is independent of the data analysis procedure itself. Thus, the L1 triggers generated in the aforementioned testbed can be used for our proposed algorithm based on collaborative local data analysis, and executed through the wireless system infrastructure.

*3) Performance Metrics and Input Parameters:* To study the performance of our algorithm in the presence of communication failures we consider two input parameters: *broadcast frequency* and *link loss probability* $p$. For simplicity, we assume that the broadcast frequency is $1Hz$, $i.e.$ each station broadcasts once per second. We measure the performance in terms of number of false negatives, false positives, and the amount of bandwidth imposed by the algorithm on the system layer.

### B. Results

We first validate our collaborative local data analysis approach. To that end, we executed our algorithm in an environment with perfect communication medium. Each station indeed observed the same L3 triggers as the ones obtained

| Triggers Type | # Triggers |
|---|---|
| L1 | 583,455 |
| L2 | 323,099 |
| L3 | 218,202 |
| DR(L3) | 93,932 |

Fig. 2: Filtering capability of the algorithm.

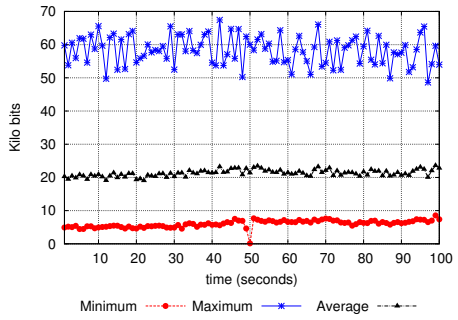



Fig. 3: A temporal view of the bandwidth production by the algorithm.

through the existing centralized approach, described earlier. This means that our algorithm behaves correctly.

We evaluate the performance of our distributed algorithm with respect to the number of false negatives, false positives, and the amount of bandwidth imposed by the algorithm on the system layer. The aim is to reduce both false negatives and false positives. At the same time, for a certain level of false negatives and false positives, we want to anlyze the *maximum* bandwidth requirement of the algorithm.

First we analyse the performance of the algorithm assuming perfect communication among the stations. The assumption is made for two reasons. First, to demonstrate the potential of local data analysis to filter out relevant data. Second, the obtained performance will be used as reference for comparison. Figure 2 depicts the filtering capability of the algorithm. Considering the whole network of stations as a single entity, we see that the network observes a huge number of L1 triggers. Note that these L1 triggers have been observed over a period of 100 seconds. The algorithm processes these triggers in-network. As a result, only one third of the triggers are promoted to L3 triggers. By definition, an L3 trigger is called an *event of interest*. Therefore, the L1 triggers at a station that were not promoted to L3 triggers are discarded locally. Each station furthermore applies a so-called direction reconstruction (DR) filter [1] [11] to its L3 triggers. The filter discards those L3 triggers whose corresponding directions point to the horizon. The *zenith* angle in the range $90 \pm 5$ is considered as horizon. We see that the number of relevant triggers to be sent to the CRS are further reduced by applying the $DR$ filter. So, under the assumption of perfect communication, the algorithm is able to discard approximately $83\%$ of the triggers locally.

The algorithm demands a certain capacity from the *system layer* to process the triggers occurred at stations. The most crucial is the bandwidth required by the algorithm. To that end, we measure the bandwidth (imposed by each station on the system layer) in consecutive time windows each of length $Tw$. We compute the maximum, minimum, and average bandwidth required per station during $Tw$. For simplicity, we assume $Tw$=1 second. Figure 3 shows the temporal dynamics of these metrics. We see that the maximum is far away from the corresponding average. In general, there are two possibilities for the large gap between maximum and average. First, there may be a specific station continuously triggering with high rate and generating relatively higher amount of data. Consequently, each time the maximum for $Tw$ is contributed by this particular station. A second possibility is that different stations during different $Tw$ produce burst data that pushes the maximum away from average. In our specific trace, we noticed that there are a few neighboring stations that trigger with high rate and push the maximum upward. In principle, the system layer should be able to absorb the imposed maximum bandwidth irrespective of the underlying cause.

Next we analyse the performance of the algorithm assuming that communication links at the system layer may fail with a probability $p$. Due to link failures a station may not be able to receive enough information from its neighbors to decide about its local L1 triggers. In this situation, discarding those L1 triggers that were not promoted to L3 triggers due to lack of information will give rise to false negatives. Our aim is to keep the number of false negatives low. To this end, we use our compensation algorithms that basically produce message redundancy in the network; thus increasing the chances of message delivery. To establish a basis for comparison, a fixed rebroadcast is used where each message is broadcast twice. Thereby, it is assumed that a link can lose a message with $p = 0.5$. In addition to fixed rebroadcast, adaptive rebroadcast is used where each station maintains a Link Quality Estimator (LQE) [15]. A message is rebroadcast only if the weakest link quality is above $50\%$ and below $100\%$. The links with quality below $50\%$ are considered dead, therefore, rebroadcast is abandoned. We examine the impact of compensation algorithms on performance. More specifically, three different cases are considered. First, the core algorithm is executed without any compensation algorithm; the so-called base case. Second, the core algorithm is executed with fixed rebroadcast. Third, the core algorithm is executed with adaptive rebroadcast. All the three cases are repeated with various link loss probabilities. We are interested to see the extent to which the compensation algorithms compensate the link failures by reducing the number of false negatives.

Figure 4 shows a comparison of the three cases for a range of message-loss probabilities. For $p = 0$ all the three cases have no false negatives because of no link failures. For $p = 1$ all the three cases have $100\%$ false negatives for the obvious reason that all messages are lost by the system layer and none of the stations is able to compute L3 triggers. The cases with message redundancy produce fewer number of false negatives than the base case. This shows the effectiveness of the compensation mechanisms in improving the accuracy of

[1]The station reconstructs direction of the signal that caused the L3 trigger. The direction is expressed as a tuple of zenith and azimuth angles.

the core algorithm. The adaptive case performs better than base case only when $p < 0.5$. This is because of the way the adaptive rebroadcast works. For $p \geqslant 0.5$ the link qualities computed through the LQE are mostly below $50\%$. Since rebroadcasts are abandoned for link quality below $50\%$, the adaptive case behaves similar to base case for $p \geqslant 0.5$. The fixed rebroadcast is expected to produce higher redundancy than the adaptive rebroadcast. The result is that it outperforms the adaptive case by producing fewer false negatives.

We examine the worst-case bandwidth requirements of our compensation algorithms by computing the maximum bandwidth required per station per time window $Tw$. For simplicity we assume $Tw = 1$ second. Figure 5 depicts the effect of compensation algorithms on the worst-case bandwidth requirements. For $p = 0$ the maximum bandwidth requirements for the adaptive case is the same as the base case. The reason is that due to no message loss the adaptive case does not rebroadcast messages and produces exactly the same maximum bandwidth as the base case. For $p = 0$, the fixed case produces bandwidth that is double of the base case. The reason is that the fixed case broadcasts every message twice. Due to no loss of messages the produced bandwidth is exactly double of the corresponding base case. A similar behavior can be observed for $p = 1$. Every station is unable to compute L3 triggers due to unavailability of information from its neighbors. So a station broadcasts only its local L1 triggers. There are no rebroadcasts in the adaptive case because for $p = 1$ the link quality remains zero and rebroadcast is abandoned. The maximum bandwidth requirement of the base case is less than both the fixed case and the adaptive case. However, the base case produces comparatively more false negatives. On the other hand, the fixed case imposes the highest bandwidth requirement but produces the least number of false negatives. The adaptive case tries to reduce bandwidth production by selectively rebroadcasting. The adaptive case reduces bandwidth compared to the fixed case but at the cost of producing higher number of false negatives than the fixed case. We see that more accurate event detection requires additional bandwidth. The system layer is required to meet the worst case bandwidth requirement in order to keep the number of false negatives within a certain limit.

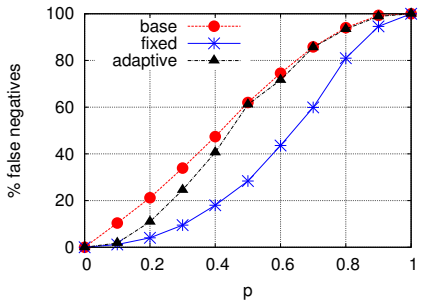
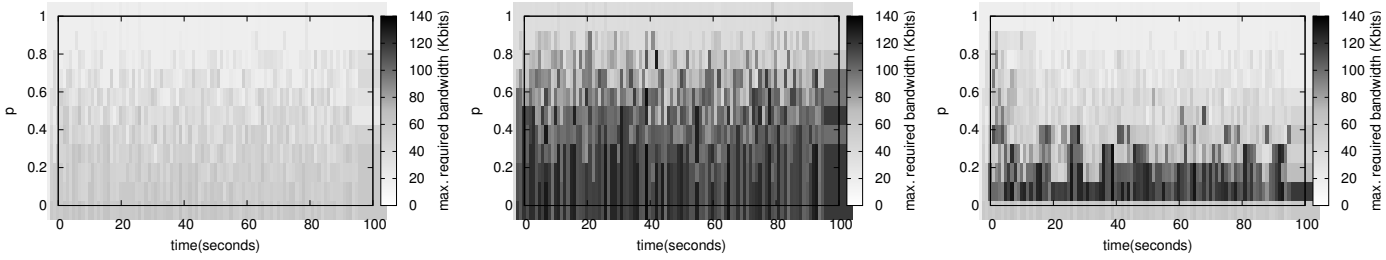



Fig. 4: The effect of compensation algorithms on false negatives.

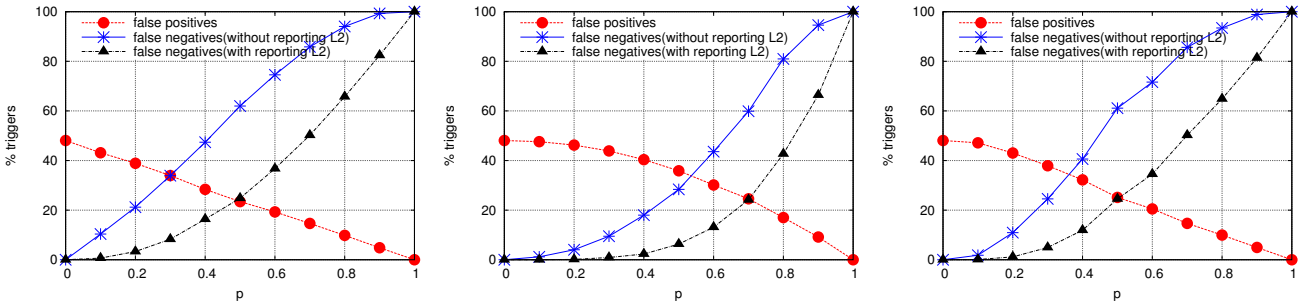
Another approach to minimize false negatives is the way the algorithm handles undecided triggers local to a station. We consider a system layer with faulty communication links. Therefore, it is likely that a station may not be able to receive sufficient information from its neighbors. Consequently, some of the local triggers (including L1 and L2) at the station may remain undecided. There are three different options to handle these undecided triggers. First, discarding all the undecided triggers, a station may falsely discard many triggers. This will lead to a high number of false negatives. Second, reporting all undecided triggers as false positives will exclude the possibility of false negatives but this will push the number of false positives to a maximum. In fact, this option is equivalent to reporting everything to the CRS. Third, the algorithm reports only undecided L2 triggers to the CRS. There are two assumptions underlying this option. First, an L2 trigger is an L3 trigger but due to communication failures it was not promoted to L3 trigger. In case the assumption is correct, false negatives will reduce without an increase in false positives. Second, it is assumed that an undecided L1 trigger is random noise and must be discarded. Again, if the assumption is correct then neither false positives nor false negatives will increase. Otherwise a node may falsely discard the L1 trigger.

Figure 6 depicts the effect of reporting undecided L2 triggers on the performance of our three cases: base, fixed and




(a) The core algorithm without rebroadcasts. (b) The fixed rebroadcasts. (c) The adaptive rebroadcasts.

Fig. 5: The impact of compensation algorithms on maximum bandwidth production.




(a) The core algorithm without rebroadcasts. (b) The fixed rebroadcasts. (c) The adaptive rebroadcasts.

Fig. 6: The effect of reporting undecided L2 triggers to the CRS.

adaptive for various link loss probabilities. For $p = 0$, all cases have the same number of false positives. The reason is that for $p = 0$, the communication is reliable and all possible L3 triggers are successfully computed. The L2 triggers reported in this case are definitely noise. For $p = 1$, all cases have no false positives. By definition, a false positive must be an L2 trigger. Since, for $p = 1$, all messages are lost; no station is able to compute an L2 trigger. Therefore, the number of false positives drops to zero. On the other hand, we see that the number of false negatives is maximum; all undecided L1 triggers are falsely discarded. The decline in false positives with increase in $p$ is due to the fact that fewer L2 triggers are computed in more lossy environment. For that reason, we see a sharp decline in false positives for the base case. In general, we see that by reporting undecided L2 triggers the number of false negatives decrease. Moreover, the rate of decrease in false negatives is higher in cases where the core algorithm is executed in combination with some compensation algorithm. However, the higher decrease in false negatives is at the cost of higher number of false positives in the corresponding cases.

## V. Conclusion

This study has uncovered that collaborative local data analysis using wireless communication is the only geographically scalable solution for high-energy cosmic ray detection. However, it suffers from false negatives due to unreliable wireless communication. Since high-energy cosmic rays are extremely rare, false negatives are unacceptable. To reduce false negatives, we evaluated two approaches: message redundancy and reporting partially aggregated data (L2 triggers). However, message redundancy requires additional bandwidth. Similarly, reporting L2 triggers causes false positives which are also resource consuming. We notice that under ideal circumstances, the bandwidth requirements exceed what many networks solutions (e.g. Zigbee) can currently provide. We have not yet considered saving energy by going into duty-cycle mode or increasing the neighborhood to increase robustness at the cost of sharing the available bandwidth among more neighbors. In short, we have to be careful on the selection of system layer technology used for wireless communication. A challenging task for further research is to evaluate our proposed approach under real radio models with bandwidth constraints and more detailed message-loss patterns in the network.

## References


[1] J. Gehrke and S. Madden, "Query processing in sensor networks," *IEEE Pervasive Computing*, vol. 3, pp. 46–55, 2004.

[2] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 131–146, 2002.

[3] F. Martincic and L. Schwiebert, "Distributed Event Detection in Sensor Networks," in *Proc. Conf. Systems and Networks Communication (ICSNC)*. IEEE Computer Society, 2006, pp. 43–48.

[4] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proc. Symp. on Operating systems design and implementation (OSDI)*. USENIX Association, 2006, pp. 381–396.

[5] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network," in *Proc. European Workshop on Wireless Sensor Networks (EWSN)*. IEEE, 2005, pp. 108–120.

[6] G. Wittenburg, N. Dziengel, C. Wartenburger, and J. Schiller, "A system for distributed event detection in wireless sensor networks," in *Proc. Conf. Information Processing in Sensor Networks*. ACM, 2010, pp. 94–104.

[7] D. Müller and M. Ave and P. J. Boyle and F. Gahbauer and C. Höppner and J. Hrandel A and M. Ichimura B and Müller and A. Romero-wolf, "The TRACER Project: Instrument Concept, Balloon Flights, and Analysis Procedures," in *Proc. Int. Cosmic Ray Conf.*, ser. OG part 1, vol. 2, 2007, pp. 83–86.

[8] "The Cosmic Ray Energetics and Mass (CREAM)," http://cosmicray.umd.edu/cream/.

[9] "Alpha Magnetic Spectrometer," http://ams-02project.jsc.nasa.gov/html/Projectpage.htm.

[10] C. Aramo, "Ultrahigh energy cosmic rays detection," *AIP Conference Proceedings*, vol. 794, no. 1, pp. 240–243, 2005.

[11] T. Huege, "Radio detection of cosmic rays in the Pierre Auger Observatory," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 617, pp. 484–487, 2010.

[12] J. Blümer, R. Engel, and J. R. Hörandel, "Cosmic rays from the knee to the highest energies," *Progress in Particle and Nuclear Physics*, vol. 63, no. 2, pp. 293–338, 2009.

[13] J. L. Kelley, "For the Pierre Auger Observatory Collaboration," in *Proc. Int. Cosmic Ray Conf.*, no. arXiv:1107.4807, 2011.

[14] A. Varga, "OMNeT++ Discrete Event Simulation System." [Online]. Available: http://www.omnetpp.org

[15] A. Woo, T. Tong, and D. E. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *Proc. Conf. on Embedded Networked Sensor Systems (SenSys)*. ACM, 2003, pp. 14–27.