Significance-Based Failure and Interference Detection in Data Streams

Nickolas J.G. Falkner and Quan Z. Sheng

School of Computer Science, The University of Adelaide
Adelaide, SA 5005, Australia
{jnick,qsheng}@cs.adelaide.edu.au

Abstract. Detecting the failure of a data stream is relatively easy when the stream is continually full of data. The transfer of large amounts of data allows for the simple detection of interference, whether accidental or malicious. However, during interference, data transmission can become irregular, rather than smooth. When the traffic is intermittent, it is harder to detect when failure has occurred and may lead to an application at the receiving end requesting retransmission or disconnecting. Request retransmission places additional load on a system and disconnection can lead to unnecessary reversion to a checkpointed database, before reconnecting and reissuing the same request or response. In this paper, we model the traffic in data streams as a set of significant events, with an arrival rate distributed with a Poisson distribution. Once an arrival rate has been determined, over-time, or lost, events can be determined with a greater chance of reliability. This model also allows for the alteration of the rate parameter to reflect changes in the system and provides support for multiple levels of data aggregation. One significant benefit of the Poisson-based model is that transmission events can be deliberately manipulated in time to provide a steganographic channel that confirms sender/receiver identity.

1 Introduction

The extensive use of sensor networks and distributed data gathering systems has increased both the rate and quantity of data that is delivered to receiving and processing nodes. Rather than processing a finite number of static records at a computationally-convenient time, data streams represent the fluctuating and potentially continuous flow of data from dynamic sources [1,2,3]. Data streams provide a rich and challenging source of data, with pattern identification and structure extraction providing important business knowledge for scientific, financial and business applications.

Several challenges occur when processing a data stream. Current data stream management techniques focus on a continuously generated stream of information and project analysis windows onto this stream, based on time intervals. While this works for a large number of applications, applications that wish to minimise onwards transmission based on the importance of data values may produce a data stream that appears discontinuous or fragmented. While a stream may be seen to never terminate, with a continuous flow of data, at a given time there may be no transmission activity within the data stream. Although the data stream may represent an abstract continuous data feed, it is implemented as a flow of data packets. Given that a data stream is composed of individual data packets, all data streams are, at a network level, non-continuous but, within the database and sensing world, the *stream* is considered to be a continuous source of data from establishment to disconnection. Analysing this stream often requires a Data Stream Management System (DSMS) [4] that will place a set of expectations upon the performance characteristics of the data stream. The DSMS then provides an ability to query the stream, in an irregular, periodic or continuous manner. While data traffic on two data streams can be irregularly spaced or presented out-of-order [5,6], there is, however, an assumption that packets in a single stream will arrive at an, explicitly or implicitly defined, arrival rate that meets the expectations of both producer and consumer. Some challenges that we must address to provide the widest applicable model for data streams include:

- **Relaxing rigid time constraints:** The assumption that a data stream is both continuous and regularly periodic is a useful assumption, except where it is incorrect. For wireless sensor networks, with low energy budgets and aperiodic transmission profiles, a requirement to conform to a near-continuous transmission schedule will drain their resources rapidly. A requirement to conform to centralised time constraints places a synchrony burden on the system that requires specialised hardware or time poll updates to minimise drift.
- **Managing significant data as a separate class of values:** In any stream of data, some values will be of more significance than others. We wish to be able to process significant events in a timely fashion, potentially in a large collection of insignificant values. Without a classification model that allows the separation of the values in a data stream into distinct sets of significant events, we must deal with all values in the stream at the same level of priority for analysis and onwards transmission.
- **Failure detection without constant polling:** We wish to be able to detect failure and, preferably, interference or tampering without having to send a continuous set of data packets. Unless they are significant, continuous data packets only add to the amount of traffic that must be discarded at the sink, reducing effective bandwidth and consuming resources.

Consider two important applications for data stream interpretation: determining similar subsequences between data streams and determining skyline elements within a data stream. The subsequence problem requires the identification of elements of the data stream that are sufficiently characteristic to be identified in another stream, and the skyline problem requires us to find dominant elements in the stream, either across the entire stream or within a sliding window. In both cases, we are moving beyond a naive description of the stream it is now a stream of significant events, where this significance and its definition are key to our ability to mine the data stream while it is still being produced. Without an ongoing, and potentially evolvable, definition of significance, we cannot begin processing until we are sure that we have received all of the data and can now conduct our comparisons or selections with absolute certainty. This is, obviously, not a workable solution.

Our contribution is to provide a statistically-based model that can provide multiple views of the same data stream, with significance thresholds and arrival expectations defined for each view. By communicating expected rate information and comparing with actual rate information, we can detect failure and interference, and, in addition, exploit the statistical characteristics to provide an out-of-band communications channel. In this paper, we will provide a description of a significant event data stream that defines the nature of significance, the impact of the significance on the interpretation of the stream, and the expectation of the inter-arrival time in a way that allows the consumer and producer to agree upon a given rate, without being required to enforce a synchronous schedule.

The remaining sections of the paper are organized as follows. Section 2 presents the related work in this area. Section 3 introduces our model and provides the key definitions. Section 4 presents the failure detection model. The Out-of-band encoding mechanism is described in Section 5, with the results of experiments conducted to verify the work presented in Section 6. Finally, Section 7 provides some concluding remarks.

2 Related Work

Within data stream research, there are many examples of the application of an implied significant event stream over a continuous data stream. Dynamic Time Warping (DTW) [5] allows for the measurement of the similarity of two data sequences by allowing for a non-linear warping of the time scale on the measurements to provide a measurement of similarity between distinct events, regardless of their position in the data sequence. This does not, however, provide any estimates as to the constraining window that contains events of likely similarity. Other work has used Poisson-distributed test data to measure performance with well-established characteristics [7,8], but does not separate these Poisson streams for analysis, nor take the cumulative stream characteristics into account when multiple streams merge.

Streaming pattern discovery in multiple time-series (SPIRIT) [6] is an approach that employs principal component analysis to allow the location of correlations and hidden variables that are influencing the reported readings. SPIRIT is designed to allow the rapid incorporation of received events into dynamic weights, rather than detecting when a predicted event has not been received. SPIRIT can provide a reliable estimate by applying forecasting techniques to the hidden variables derived from the data stream and can use these to estimate missing data in the x_t data set, based on the x_{t-1} data set. However, this assumes continuous and regular data intervals. SPIRIT assumes periodicity in the data stream, rather than an irregular data stream or a data stream where the significant events are not spaced regularly in time.

StatStream [9] allows for the real-time analysis of large volumes of data, employing the Discrete Fourier Transform and a three-tiered time interval hierarchy. StatStream does assume that the data stream cannot be regarded as having a terminating point, and works on the data stream continuously, in order to meet real-time requirements. To be explicit, a data stream is regarded as a sequence, rather than a set. StatStream provides a basis for stating that any statistic present in the data stream at time t will be reported at time $t+v$, where v is a constant and is independent of the size and duration of the stream [9]. StatStream establishes three time periods: i) timepoints - the system quantum, ii) basic window - a consecutive subsequence of time points which comprise a digest, and iii) sliding window - a user-defined consecutive subsequence of basic windows that will form the basis for the time period over which a query may be executed.

While this provides a great deal of flexibility in dealing with intervals, StatStream expects to have at least one value per timepoint and, if one is not present, an interpolated value is used. This does not accommodate two possibilities. The first is that there is a reading but it is not sufficiently important to report at that time, where the system is filtering the result. The second is that there is no sensor reading to report at that point because the size of the time interval and the expected number of events leads to an event/interval ratio less than 1. More importantly, interpolation in the face of missing data may insert a false reading into the network. To explain this, it is first important to realise that, in the event of multiple values being reported in one timepoint, StatStream will provide a summary value. The reported arrival of an event in an adjacent interval can result in the preceding time interval being reported incorrectly, with the next interval providing a summary value that is also artificially high. While the synthesis of summary values will, over time, produce the same stream characteristics, there is no clear indication that an irregularity has occurred, nor can action take place to rectify the mistake. This summarisation can also obscure the point where a value, or set of values, has crossed the significance threshold.

There is a great deal of existing work in the field of sensor networks pertaining to more efficient use of resources through filtering, the discarding of insignificant data, and aggregation of significant data for more efficient upstream transmission. There is very little work that addresses the modelling of the implicit stream of significant events that these networks generate. Gu et al. [10] discuss a lightweight classification scheme for wireless sensor networks employing a hierarchical classification architecture but do not address statistical detection of node failure. Solis and Obraczka [11] discuss several aggregation mechanisms for power-efficiency but use temporal aggregation mechanisms and do not address node failure. Ye et al. [12] propose a statistical filtering process for injected false data but do not address statistical mechanisms for determining sensor operation.

3 Traffic Modelling in Intermittent Data Streams

Formally, a data stream is an ordered pair (s, δ) where s is a sequence of tuples of the form $(\xi_0, \xi_1 ... \xi_N)$, representing the data values contained in a single data packet, and δ is a sequence of time intervals where $\forall i, \delta_i > 0$.

In a regular, continuous data stream, the deviation from the average arrival time for each i is much smaller than the average arrival time (Equation 1). When data starts to arrive in an irregular fashion, the deviation can be of the same order of magnitude as the average time with a multiplier k, where k may be a fixed bound or may vary with time (Equation 2). The interpacket time interval, δ_i, must be sufficiently small that the client's expectation of continuity is not contradicted, otherwise the stream no longer appears continuous, i.e., it appears to be composed of irregular packets.

$$|\delta_{average} - \delta_i| < \epsilon, \forall i, \epsilon \ll \delta_{average} \quad (1)$$

$$|\delta_{average} - \delta_i| <= k\epsilon, \forall i, \epsilon \approx |\delta_{average}| \quad (2)$$

A user's expectation of continuity of a stream is only satisfied if k is sufficiently small that the δ_i values have an acceptable upper bound, δ_{max} (Equation 3). However, determining a reasonable expectation for δ_{max} is difficult, where data inter-arrival is irregular, as it may require a great deal of observation that is potentially only valid for one producer and consumer pair, or for one connection at a given time. This problem becomes more complex when we seek to place additional structure into the ongoing interpretation of a data stream.

$$\forall i, \delta_i < \delta_{max}, \delta_{max} >= \delta_{average} \tag{3}$$

3.1 Definition of the Model

Definition 1. *A significant event data stream, SED, is an ordered tuple $(\mathcal{L}, \mathcal{E}, \Delta_{\mathcal{L}}, \lambda_{\mathcal{L}})$ where:*

- *A tuple $(\xi_0, \xi_1 ... \xi_N)$ in the sequence s is significant if the removal or alteration of the tuple will have a significant and discrete impact on the result of a specified computation that depends on s.*
- *$\mathcal{L}$ is a unique label, identifying this SED.*
- *$\mathcal{E}$ is a sequence of tuples, such that $\mathcal{E} \subseteq s$ and all tuples in $\mathcal{E}$ are significant.*
- *$\Delta_{\mathcal{L}}$ is a sequence of time intervals where $\forall i, \Delta_{\mathcal{L}} > 0$ and $\Delta_{\mathcal{L}} \sim Pois(\lambda_{\mathcal{L}})$.*
- *$\lambda_{\mathcal{L}}$ is the expected arrival rate of $\mathcal{E}_i \in \mathcal{E}$.*
- *Any SED is a subsequence of an existing data stream, $\mathcal{S} : (s, \delta)$ where $(s_i, \delta_i) \in (\mathcal{E}, \Delta) \not\Rightarrow (s_{i+1}, \delta_{i+1}) \in (\mathcal{E}, \Delta)$.* □

Definition 2. *A SED Implementing Model (SEDIM) for a data stream is defined as follows:*

- *The data stream $\mathcal{S}$ is an ordered pair (s, δ) as defined previously,*
- *Within $\mathcal{S}$, there exists a set of significant event data streams, $\hat{\mathcal{S}} : (\hat{\mathcal{S}}_0 ... \hat{\mathcal{S}}_i), i \geq 0$.*
- *A significant event $\mathcal{E}$ in $(\hat{\mathcal{S}}_i)$ is defined such $\forall \mathcal{E}_i \in \mathcal{E}$, $\mathcal{V}_{lower_i} \leq \mathcal{E}_i \leq \mathcal{V}_{upper_i}$, where $\mathcal{V}_{upper_i}$ and $\mathcal{V}_{lower_i}$ represent the bounds of significance for the SED $\hat{\mathcal{S}}_i$.*
- *The set $\Lambda : (\lambda_{\mathcal{L}_0} ... \lambda_{\mathcal{L}_n})$ is a set of independent rate parameters for the expected arrival rate of significant data in the SEDs $\hat{\mathcal{S}}$.*
- *t_{now} is defined as the current time in the system.*
- *t_{window} is defined as the time to the expiry of the current sliding window associated with a user query. An ongoing query may generate many subqueries, all of which have their own sliding window.*
- *The received set $\mathcal{R}$ is the set of all events that have been consumed in order to meet the requirements of the current query. The event $\mathcal{E}_{\mathcal{R}}$ is the last event that has been received and the event $\mathcal{E}_{\mathcal{R}+1}$ is the next event that will occur, regardless of the SED or stream that generates it. All events $\mathcal{E}_{\mathcal{R}+1}$ have an associated time interval $\Delta_{\mathcal{R}+1}$ and, by definition, $\sum_{j=0}^{\mathcal{R}+1} \Delta_j > t_{now}$.* □

3.2 Rationale

We must first justify that a data stream can be regarded as continuous, and still have the potential for irregular and insignificant data packet transfer. Many data stream applications make a similar assumption to StatStream, namely that there exists a system quantum and that events will arrive, at a roughly equal spacing, at a rate determined by the size of this quantum. For example, if the quantum is a second, the expected arrival rate is one event per second. This does, however, immediately provide the possibility of a continuous stream that can be viewed in such a way as to appear non-continuous in transmission. Consider a system with a one second timepoint and then, without changing the rate of arrival, change the timepoint to 0.1 seconds. Now the arrival rate is 1 event in every 10 timepoints and this, originally continuous and 100% utilised, data stream is now occupied 10% of the time. Considering the impact of pragmatic networking considerations on the transmission of data, given that it takes a finite non-zero time to transfer a network packet, there must exist a timepoint, $\mathcal{T}_\epsilon$, such that any data stream may be regarded non-continuous, regardless of the regularity of data transmission.

A more complex problem arises when, rather than managing regular transmission in a non-continuous data stream, we must consider the effect of irregularly spaced data, whether this is due to an absence of data or the insignificance of the data being transmitted. In the previous example, non-continuous data only constitutes a problem if the receiver fails to define their own aggregation or summary operations correctly for a given query in a sliding window. For example, if the value 10 is sent once per 10 timepoints, is the average over time 10 or 1? This will have an impact on the final answer delivered in response to the query "What is the average value over the sliding window from t_i to t_{i+x}?". As previously illustrated in the discussion of StatStream, a value that is incorrectly placed into a different timepoint can have a significant impact on the result of queries that span a subsequence, and this extends to the boundaries of sliding windows if they are purely timebased, rather than taking into account the possibility that an interval does not contain the same number of events. The query "What is the average value of the significant events received over the sliding window from t_i to t_{i+x}?" is an unambiguous query and, in the example above, would result in the value 10.

A data stream that is intermittently active may or may not have an associated sequence of significant events, given that the definition of significance is associated primarily with the consumer. However, if a significant event sequence exists within a data stream, the sequence may have a regular period that allows simple prediction of inter-event time, as the data stream can be composed of ongoing data combined with a regularly inserted significant element. It is, however, far more likely that events of significance will be more randomly distributed, unless what is being monitored for significance is naturally periodic or has been defined to be so.

We have employed the Poisson distribution to model the behaviour of significant event sequences arriving over data streams. The Poisson distribution is used where a number of discrete events occur within a given time-interval. Where an approximation can be made to the arrival rate, the Poisson distribution can be used to establish the inter-arrival time and also determine when it is likely that an event is lost or non-existent, rather than late. A significant advantage of the Poisson distribution is that, among other benefits, the combination of Poisson processes is another Poisson process with a rate parameter that is the sum of the composing processes. Changing the arrival rate allows the immediate alteration of the expectation of the arrival rate and inter-arrival rate of future events, without requiring the storage of previous event and time pairs.

The Poisson distribution, for rate λ, has a mean of λ. Thus, once an arrival rate has been established for a given interval, λ, we would expect λ events per interval. Given that we wish to be able to predict arrival and, hence, failure to arrive, we need to be able to predict the inter-arrival time. If the number of arrivals in a time interval $[t_i...t_{i+x}]$ follows the Poisson distribution, with rate parameter λ, then the lengths of the inter-arrival times are defined as $\mathcal{I} \sim Exponential(\lambda)$, with a mean inter-arrival time of $\frac{1}{\lambda}$. This is an important result as it clearly shows that the Poisson distribution will correctly model a continuous, regularly spaced data stream but it also gives us the ability to model a non-continuous, irregularly spaced data stream.

3.3 An Example

Wireless sensor network applications may employ multiple sensors in an observation domain, on the same node or by combining the results of several nodes [13,14]. This example illustrates a surveillance application, with a widespread WSN constructed of nodes that employ vibration sensors, photosensors and acoustic sensors. This WSN is spread over a large geographical range and has a lifespan measured in months. Maintenance is limited, due to the cost and time involved in travelling to the sensor nodes. With the sensor nodes ground-mounted, the vibration sensors report ground movement in the vicinity of the sensor, photosensors report an interruption to a light beam projected at the node from a nearby position and the acoustic sensors provide a measure of the acoustic pressure in the region. These three sensors display the range of possible sensor event generation. The vibration sensor will be continuously reading small vibrations in the ground, the photosensor is effectively boolean in that the light beam is either striking it or it is interrupted, and the acoustic sensor is more likely to manifest a combination of the previous two, as sound pressures can easily drop below the detectable level but have a continuous distribution once detected.

At each sensor node, a sequence of tuples, s, is generated each time transmission occurs. The sensor readings are only a subset of this tuple as the $(\xi_0...\xi_N)$ in s also include information such as source and destination, as well as any other system-specific information. The sensor reading tuple takes the form $(\eta_0, \eta_1, \eta_2, \eta_3)$, where:

- η_0 is the timepoint at which the data was sensed,
- η_1 is the vibration reading,
- η_2 is the photosensor reading,
- η_3 is the acoustic pressure

The sensor nodes employ both filtering and aggregation to limit upstream transmission but, given their remote location, each node must transmit sufficiently often to prevent unneeded maintenance visits [11]. In this example, the nodes employ local significance filtering, but no aggregation. Aggregation is carried out at a regional level, with filtering also employed on the aggregates where necessary, and may take a number of forms [15,16]. Such aggregation and filtering is vital because there is a maximum capacity

SED	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}
VIB:	10	10	4	6	4		100		10	10	4	6	4		100	
PS:							1		1						1	1
AP:					120			120								
INSIG:	2	2	2	2	1	3	1	3	1	2	2	1	2	3	1	2

Fig. 1. Sample data stream decomposition

for a given sensor network, not just because of the information that each node wishes to send, because of the requirement for nodes to route information for other nodes. Ultimately, any sensor network has a maximum throughout, based on the number of nodes and their available bandwidth [17]. It is essential to keep bandwidth use below this threshold.

The significance thresholds for instantaneous readings on each sensor are Sig_{vib}, Sig_{ps} and Sig_{ap}, reflecting the level that separates filtered events from unfiltered events.

Defining Significant Event Streams. Within this SEDIM, there are four SEDs. These are:

- $(VIB, \mathcal{E} : \eta_1 >= Sig_{vib}, \Delta_{Vib}, \lambda_{Vib})$
- $(PS, \mathcal{E} : \eta_2 >= Sig_{ps}, \Delta_{ps}, \lambda_{ps})$
- $(AP, \mathcal{E} : \eta_3 >= Sig_{ap}, \Delta_{ap}, \lambda_{ap})$
- $(INSIG, \mathcal{E} \notin [VIB, PS, AP], \Delta_{insig}, \lambda_{insig})$

and the corresponding data model is $\mathcal{S}_{\mathcal{E}}$, where

- $\mathcal{S}_{\mathcal{E}} : [VIB, PS, AP, INSIG]$
- $\Lambda : [\lambda_{Vib}, \lambda_{ps}, \lambda_{ap}, \lambda_{insig}]$

Given that the events in $\mathcal{S}_{\mathcal{E}}$ are the tuples $(\xi_0 ... \xi_N)$, where each ξ is composed of transmission headers and footers and the sensor reading tuple $(\eta_0, \eta_1, \eta_2, \eta_3)$, we can now model an individual SED as a set of events that meet the SED criteria. This allows us to provide sample data streams for each SED, which would be interspersed in the true data stream, and analyse them separately at any node that can carry out filtering, aggregation and analysis.

The Sample Streams. This example presents tuples containing the sample data detected by the sensor node. Each tuple entry represents one poll of the sensors and all tuples are defined to be in synchrony. Each tuple entry is a timestamp (η_0) and the associated value (i.e., η_1, η_2, or η_3). The INSIG SED contains the cardinality of insignificant values, rather than the values themselves, although this is an implementation decision. Figure 1 shows a decomposed data stream, with the values allocated to the SEDs and insignificant values shown as cardinalities in the INSIG SED.

If we define t_0 as the time at the start of interval Δ_i and t_{now} as t_{15}, then we can regard each t_i as a discrete tick within the time interval. For the purposes of the example, the range $[t_0..t_{now}]$ is divided between time intervals Δ_i and Δ_{i+1}.

The estimated arrival rates (EAR), defined by previous observation and established as system baselines, for each SED are given in Figure 2 along with the Time Interval

SED	EAR	TIAAR (Δ_i, Δ_{i+1})	AAR (Δ_i, Δ_{i+1})	Interval
VIB	6	(6,6)	(6,6)	8 ticks
PS	2	(1,3)	(2,2)	8 ticks
AP	1	(2,0)	(1,1)	8 ticks

Fig. 2. Estimated arrival rates

SED	Transmission times (ticks)
VIB	t_8 and t_{16}
PS	t_9 and t_{16}
AP	t_{16} (first transmission)

Fig. 3. Transmission times

Aligned Arrival Rate (TIAAR), the Actual Arrival Rate (AAR) and the interval over which the rate is measured. TIAAR shows the rate that would be returned by a naive interpretation of the data stream on strict time boundaries, while AAR gives the arrival rate adjusted for the relaxation implicit in accepting data within the Poisson noise. Both TIAAR and AAR are given as a pair of values, one for each time interval.

The INSIG stream is continuous, as the cardinality of events is continuously generated. However, this SED is for internal reference only, and is not transmitted to other nodes (although a summary may be requested by a superior node).

The AAR values match the EAR values because 'late' events can be accepted within $\sqrt{EAR}$, in terms of the fraction of the previous interval that the collection window is still considered to be opened. Importantly, events accepted as part of a previous window cannot be counted as part of a subsequent window and, in the case of SED_{PS}, the second interval has the range $[t_9..t_{15}]$, as the t_8 tick has been included in the previous window. While not shown in Figure 1, t_{16} is the next tick that will occur after t_{15} and denotes the start of the next interval.

The delay introduced by this scheme can be seen by displaying the transmission times, as a given tick t_i, where the tuple is encapsulated and sent to the network as a significant packet. These transmissions times are shown in Figure 3.

The AP delay is the most significant, as we must wait an entire interval to ensure that the event at t_7 is an advanced event from $\Delta_i + 1$ (within the noise parameter), rather than an additional event from interval Δ_i. As the λ_{AP} is 1, and $\sqrt{1}$ is 1, we can wait an entire interval before receiving the packed that was supposed to have occurred in Δ_i. However, this is the worst case situation - the maximum delay inserted is one interval length in the rare situation that we are only expecting one event per interval.

4 Failure Detection

Failure detection in a continuous, regular data stream is relatively straight forward: the data stops. In a SED, a missing event poses a more complex problem as there are different possibilities:

1. Events are still arriving but are below the significance threshold of a given SED.
2. An event will be sent but it will arrive slightly after the deadline.
3. The event is actually lost.

In terms of the data stream $\mathcal{S}$, we can describe each of these possibilities as equations.

$$\exists\hat{\mathcal{S}}_i : \forall\mathcal{E}_i \in \mathcal{E}, \mathcal{E}_i \notin [\mathcal{V}_{lower_i}...\mathcal{V}_{upper_i}], \varDelta_i >= t_{now} \tag{4}$$

$$\exists\hat{\mathcal{S}}_i : \exists(\mathcal{E}_{\mathcal{R}+1}, \varDelta_{\mathcal{R}+1}), \varDelta_{\mathcal{R}+1} > t_{window} \tag{5}$$

$$\nexists\hat{\mathcal{S}}_i : \exists(\mathcal{E}_{\mathcal{R}+1}, \varDelta_{\mathcal{R}+1}) \tag{6}$$

In $\mathcal{S}$, equation 4 only constitutes failure in the SED that has the restrictive range of significance. In this case, it does not constitute a stream failure but indicates that no significant events are arriving. This may indicate that there are no significant events to report or that there is a mis-reporting of events. There are several possible reactions:

1. After a given time, adjust the range of significance to reflect increased knowledge of the data stream contents. This is referred to as *rate relaxation*.
2. The producer can generate a system message that provides evidence that, should a significant event occur, it will be generated and passed on. We refer to this as *significance exchange*.
3. If test equipment is in place, an artificially significant event is generated and sent upstream, to be discarded by the consumer. We refer to this as *significance verification*.
4. Ignore it and drop this SED from $\mathcal{S}$. This may also be considered equivalent to a rate relaxation to a parameter of zero - we expect nothing significant to occur in an interval.
5. Report it.

Rate relaxation increases the time over which events may be detected. As the Poisson distribution may only take integer valued parameters, the minimum non-zero rate is one event per interval. Once the rate has been relaxed to one event per interval, the only further relaxation possible is the extension of the interval and this is carried out by doubling the interval size, to a maximum value of 1 calendar year, although it may be smaller for a more short-lived system. Once relaxation has occurred to the occurrence and time limit, any further relaxation will set the rate parameter to zero, effectively terminating the expectation of arrivals in this SED. This is equivalent to reaction 4.

Significance exchange requires both producer and consumer to be able to exchange meta-values that describe the context or value ranges expected for the values. XML, provided that there is a contextual basis such as RDF or OWL-XML in place describing the shared context, may be used to exchange system messages that are not interpreted as events but contain information that confirm what both parties consider significance to be. These messages can piggy back onto events, if the events are wrapped in XML and are tagged by the producer as significant or insignificant. However, by choosing the lowest value $x : x > \mathcal{V}_{lower_i}$ and sending this as a test, the producer clearly indicates where the threshold is.

Significance verification requires that the producer be capable of injecting a significant event into the data stream and reporting on it, while labelling it in such a way that it is not treated as a significant event elsewhere. This is also a system, or test, message but, instead of simulating event handling, the test event is injected prior to detection. This requires a comprehensive test harness if physical sensors are being employed. Where data is not being acquired directly through physical sensors, we can separate the network reading component and data processing component of the producer and insert a data injection mechanism between the two. This also relies upon the ability of both producer and consumer to agree upon what constitutes a test message, and to be able to send and receive meta-values, rather than a composite stream of values, parsed from a purely structural perspective into a value stream with no type or context information.

Equation 5 may cause problems in a system with a synchronous time requirement, from the movement of values into adjacent cells potentially leading to incorrect summarisation, but is manageable within a SED. This is due to key properties of the Poisson distribution. The Poisson distribution has a mean of λ but also has a variance of λ. For a given interval, the number of observed arrivals fluctuates about the mean λ with standard deviation of $\sqrt{\lambda}$, where these fluctuations are referred to as the *Poisson noise*.

Poisson noise, or *shot noise*, describes the statistical fluctuation observed from the arrival of finite, discrete events. This effect is seen in electronic applications, photon counters and particle simulations. The Poisson noise increases as the rate of arrival increases, but at a slower rate. For any number of events sampled, where the sample has a Poisson distribution, the average number of events does not reflect that true number of events detected in that interval but the actual result will be distributed about the average with a Poisson distribution.

Similar to StatStream, we now have a time interval within which we will have been able to detect the vast majority of failures and it is a fixed time, given by the arrival rate λ and the time interval, $\mathcal{T}_{\mathcal{I}}$. We declare likely failure in Δ_i for a rate λ if, for a given interval $[\Delta_i...(\Delta_{i+1}/\sqrt{\lambda})]$, the cardinality of events $\mathcal{E}_i$ in the interval is less than λ.

$$Fail(\Delta_i, \lambda) : card(\mathcal{E}_i \in [\Delta_i...(\Delta_{i+1}/\sqrt{\lambda})]) < \lambda \qquad (7)$$

We have now presented the way of dealing with possibilities contained in equations 5 and 6. If an event is merely delayed, waiting for a pre-determined period beyond the original deadline will capture the event and there is no need to handle the event as lost or carry out any adjustments to the rate parameters. However, if the event doesn't arrive, even within the noise interval, then we have successfully detected a failure event and we can now take actions as outlined in the solution to equation 4.

A useful result of equation 7 is that it is immediately apparent that the higher the rate of arrival, the shorter the proportion of an interval that is required to detect failure. In a system with a high rate of arrival and short interval, this means that failure can be detected in very short time. Conversely, a system with low rate of arrival has a correspondingly long time to failure detection.This immediately motivates the need for the use of an artificially high significance rate, employing test data to keep the rate high, while not requiring a high rate of actual events. Significance verification or significance testing can both be used to achieve these aims.

Where interest is primarily on the waiting time to a given event somewhere in the stream, the Gamma (Γ) distribution is a family of continuous probability distributions with two paramers, k and θ, that is used for modelling waiting times. With integer k, the Γ distribution is the *Erlang* distribution and is the probability distribution of the waiting time until the k-th event in a Poisson process with rate $1/\theta$. Rather than monitoring every event to determine failure, which is energy intensive, we can now observe a k-th event to determine if the waiting times are meeting our estimates.

5 Encoding Information within the Poisson Noise

One of the advantages of allowing events to be legitimately "placed" within an interval, or within the Poisson noise of the interval, is that this information can be used as an additional communications channel. In a continuous, regular data stream, varying the regularity deliberately can be used to indicate out-of-band information that allows communication between producer and consumer, without placing an additional data burden on the main channel. In a sensor network this is vital as the smaller the data stream is, the less power is consumed in all parts of the network for transmission and processing. Examples of out-of-band channel use in the real world range from the use of tone in spoken language to signify additional meaning, such as questioning, sarcasm or dubiety, when the semantic content of the written form does not need to capture this.

Encoding information within the Poisson noise requires either that the producer and consumer have a synchronous communication channel, where deliberate movement is detectable, or have the ability to embed timestamps into their data streams to indicate the point at which they planned to send the data.

In a continuous, synchronous channel environment, producer and consumer will exchange λ events per interval. The simplest encoding available in the Poisson noise is to vary the arrival time of the final event and to reduce or increase the inter-arrival time. If we encode 0 as an unlikely reduction in arrival time and 1 as an unlikely increase in inter-arrival time, then it is possible to send binary messages from producer to consumer at the rate of 1 bit per event. This, however, does rely upon the channel in question being highly reliable, with a well-defined rate of arrival.

Where we cannot assume regularity, we must use timestamps, to allow the producer to indicate to the consumer that they had planned to send the data at time t, but actually did so at time $t+x$. Whatever x is, it must still fall within the Poisson noise interval but, in this case, we now have a greater range of possible value representations available, as the reference timestamp is within the stream.

One application of this is in non-continuous, irregularly reporting low-power sensors such as wireless surveillance sensor nodes. If these nodes only report periodically, how do we know that they haven't been tampered with in the interim? One approach is to provide a pseudo-random number generator of known and very large period, or a set of pre-generated pseudo-random numbers, to both producer and consumer and to offset the producer's messages by an interval based on these numbers. This reduces both the predictability of the event transmission and provides a low-power identification mechanism for a node. By choosing a generator with a large period, determining the sequence of numbers by observations is infeasible. If the seed, or the pre-generated sequence, are physically protected within the node, then an intruder is limited in how they can bypass a node, as they cannot replace it without losing the identification out-of-band channel, and disabling the sensor will, after some interval, generate a $Fail(\Delta_i, \lambda)$ event, which will also constitute a reportable warning event.

6 Experimental Results

We used a number of simulated test environments to test classification, failure detection reliability and the time to detection of our approach. Due to space constraints, we discuss one here, a statistical simulation based on event queues, with randomised failure.

In the experiment, a SEDIM entity (SEDIMent) was constructed as a set of simulated data streams, composed of data from three sensors. The time interval was set to 3600 seconds and λ increased from 1 event per interval to 30 events per interval. The SED-based classification was used to classify the cumulative data stream into VIB, PS, AP and INSIG SEDs. Stream transmission rates were monitored at the transmitting node and at upstream nodes. The experiment was designed to test: i) the correct detection of event loss when events fell outside of the Poisson noise interval, ii) the correct estimates of the rates of individual SEDs for determining failure of an individual sensor, iii) the correct estimates of total node failure in upstream nodes, iv) maintaining node liveness through the use of injected test data, implementing significance exchange and significance verification, as described in Section 4, and finally v) node identification using variation in the Poisson noise.

Experiments were run with failure rates ranging from 0 to 50% of nodes, with 1000 iterations of each experimental configuration. The failure of an individual event to arrive was detected within one time interval with a cumulative success rate 89.2%. Where no errors occurred in contiguous intervals, the probability of success increased to 99.6%. This established successful detection of event loss. The time taken to detect failure was proportional to $\frac{1}{\lambda}$, as expected.

The removal of INSIG data streams from transmission, and the calculation of individual arrival rates for all other SED, was tested at the event injecting node and at the simulated sink node. Transmitted events maintained their TIAAR across the system and measurements of failure detection and arrival rate were consistent with the EAR for the decomposed streams across all experiments.

The simulated sink node calculated the cumulative arrival rate, Λ_{EAR}, for all simulated sensor nodes. Individual nodes maintained a counter of the number of events where INSIG was the only active SED. In any interval where INSIG was the only active SED for the entirety of the interval, a test packet was injected into the data stream, giving the EAR for the node's active SEDs and an example classification of significance. If received at the sink, the node continued to be marked live and the classification condition was checked. Total node failure was simulated by setting all significance thresholds to the maximum value and disabling test packet injection. In this case, total node failure was detected at sink nodes within $((1 + (\frac{1}{\lambda}))\delta_i)$.

Finally, node identification was tested by dividing the first Poisson noise interval within a time interval into millisecond intervals, and encoding the transmission time

Failure rate	0.00	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
MoS	0.00	0.724	2.641	15.277	99.142	194.905	271.944	337.732	393.982	446.978	498.445

Fig. 4. Measure of Suspicion metrics for increasing failure rates

as marked-up XML data, accompanying the data stream. This transmission time was moved within the available slots by employing a fixed-period rotating set of random numbers, with the sequence known to both an individual node and the sink. In the event of an event being labelled as failed, due to falling outside of the expected range, the random number that would have matched the offset is discarded at the sink. This automatically causes the rejection of the packet, should it arrive late. A *Measure of Suspicion* (MoS) is kept at the sink node, increasing monotonically for every packet that either fails to arrive, or arrives with an unexpected offset. The MoS is decreased by one for every 10 packets that arrive with the expected offset. In testing, the average MoS for the system was 0.00 for experiments without synthetic failure and all events arriving within the intervals, as expected. Figure 4 shows the MoS for higher failure rates with a thousand individual trials of one thousand events.

It is of little surprise that higher failure rates have higher levels of uncertainty as a large number of the confirmation numbers will be dropped from the queue. At failure rates above 10%, the reduction in MoS is dominated by the ongoing increase, and approaches the failure rate multiplied by the number of events, 1000 in this case. However, for low failure rate data stream producing networks, a MoS threshold of 1 allows for the detection of increasing failure rates and the possibility of compromised nodes.

7 Conclusions

We have provided a model for the flow of significant events in data streams, in terms of the rate of arrival of these events, and the distribution of these events. This model is suitable for the modelling of both regular and irregular data streams, and is event-focused, rather than time-focused. By employing this model, it is possible to detect failure in the data stream, where this failure is a failure of transmission, or an absence of significant events.

Our experimental results clearly show that this model adapts to change, as well as reducing network overhead due to i) a minimisation of polling or liveness information, and ii) a well-defined expectations of network behaviour. We have also shown a simple application of our approach, which allows additional use of a channel without having to alter time boundaries or expected arrival rates.

We have already developed three simulation models, one statistical, one software based as a node level simulation and one grounded in the TinyOS Simulation (TOSSIM) [18] environment for power consumption.We are further developing an implementation of SEDIM in a WSN environment, and ad-hoc networking environment on hand-held computing devices. The power consumption in a mobile and distributed environment is further constrained by the requirement to support ad-hoc routing protocols and route discovery. In this role, the SEDIM approach will provide significant power savings, that will allow more aggressive route discovery and maintenance, supporting stream-based communication over ad-hoc networks, as well as packet-based communication.

References

1. Golab, L., Özsu, M.T.: Issues in data stream management. SIGMOD Rec. 32(2), 5–14 (2003)
2. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining Data Streams: A Review. SIGMOD Rec. 34(2), 18–26 (2005)
3. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and Issues in Data Stream Systems. In: PODS 2002: Proc. of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 1–16. ACM, New York (2002)
4. Babu, S., Widom, J.: Continuous Queries over Data Streams. SIGMOD Rec. 30(3), 109–120 (2001)
5. Berndt, D.J., Clifford, J.: Using Dynamic Time Warping to Find Patterns in Time Series. In: AAAI 1994 Workshop on Knowledge Discovery in Databases, pp. 359–370. AAAI Press, Menlo Park (1994)
6. Papadimitriou, S., Sun, J., Faloutsos, C.: Streaming Pattern Discovery in Multiple Time-series. In: VLDB 2005: Proc. of the 31st Intl. Conference on Very Large Data Bases, pp. 697–708. ACM, New York (2005)
7. Bai, Y., Wang, F., Liu, P.: Efficiently filtering RFID data streams. In: CleanDB: The First International VLDB Workshop on Clean Databases, pp. 50–57. ACM, New York (2006)
8. Wei, Y., Son, S.H., Stankovic, J.A.: RTSTREAM: Real-Time Query Processing for Data Streams. In: 9th IEEE International Symposium on Object/component/service-oriented Real-Time Distributed Computing, pp. 141–150 (2006)
9. Zhu, Y., Shasha, D.: StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. In: VLDB 2002: Proc. of the 28th Intl. Conference on Very Large Data Bases, VLDB Endowment, pp. 358–369 (2002)
10. Gu, L., Jia, D., Vicaire, P., Yan, T., Luo, L., Tirumala, A., Cao, Q., He, T., Stankovic, J.A., Abdelzaher, T., Krogh, B.H.: Lightweight Detection and Classification for Wireless Sensor Networks in Realistic Environments. In: SenSys 2005: Proc. of the 3rd Intl. Conference on Embedded Networked Sensor Systems, pp. 205–217. ACM, New York (2005)
11. Solis, I., Obraczka, K.: In-Network Aggregation Trade-offs for Data Collection in Wireless Sensor Networks. Intl. Journal of Sensor Networks 1(3–4), 200–212 (2007)
12. Ye, F., Luo, H., Lu, S., Zhang, L.: Statistical En-Route Filtering of Injected False Data in Sensor Networks. IEEE Journal on Selected Areas in Communications 23(4), 839–850 (2005)
13. Pottie, G.J., Kaiser, W.J.: Wireless Integrated Network Sensors. Commun. ACM 43(5), 51–58 (2000)
14. Feng, J., Koushanfar, F., Potkonjak, M.: Sensor Network Architecture. Number 12 in III. In: Handbook of Sensor Networks. CRC Press, Boca Raton (2004)
15. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: a Tiny AGgregation Service for Ad-hoc Sensor Networks. SIGOPS Oper. Syst. Rev. 36(SI), 131–146 (2002)
16. Petrovic, M., Burcea, I., Jacobsen, H.A.: S-ToPSS: Semantic Toronto Publish/Subscribe System. In: VLDB 2003: Proc. of the 29th Intl. Conference on Very Large Data Bases, VLDB Endowment, pp. 1101–1104 (2003)
17. Gupta, P., Kumar, P.R.: The Capacity of Wireless Sensor Networks. IEEE Trans. Info. Theory 46(2) (2000)
18. Levis, P., Lee, N., Welsh, M., Culler, D.: TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In: SenSys '03: Proc. of the 1st Intl. Conference on Embedded Networked Sensor Systems, pp. 126–137. ACM, New York (2003)