Self-adaptive Routing in Multi-hop Sensor Networks

Themistoklis Bourdenas,[1,2] David Wood,[2] Petros Zerfos,[2] Flavio Bergamaschi,[3] Morris Sloman[1]

[1]Department of Computing, Imperial College London, UK
[2]IBM Research, T.J.Watson, NY
[3]IBM UK Laboratory, Hursley, UK

{t.bourdenas07, m.sloman}@imperial.ac.uk, {dawood, pzerfos}@us.ibm.com, flavio@uk.ibm.com

***Abstract*—Sensor networks are used for applications in monitoring harsh environments including reconnaissance and surveillance of areas that may be inaccessible to humans. Such applications depend on reliable collection, distribution and delivery of information to processing centres which may involve multi-hop wireless networks which experience disruptions in communication and exhibit packet drops, connectivity loss and congestion. Some of these faults are periodic, attributed to external, recurring factors. In this paper, we study an effective way to forecast such repetitive conditions using time-series analysis. We, further, present an application-level, autonomic routing service that adapts sensor readings routes to avoid areas in which failures or congestion are expected. A prototype system of the approach is developed based on an existing middleware solution for sensor network management. Simulation results on the performance of this approach are also presented.**

I. Introduction

Wireless sensor networks are used to monitor environments that might be impractical or unsafe for humans to enter. Areas being monitored may be too large for single hop communication to the monitoring centre, requiring collection, distribution and delivery of information that typically travels over multiple, interconnected nodes to reach processing centres. These networks may be susceptible to various communication disruptions such as connectivity loss due to unreliable links as well as packet drops due to noise on the wireless medium or high-volume of traffic overloading links and network buffers.

While many of the faults can be attributed to random events, some of them exhibit specific repeating patterns caused by periodic events in the environment, such as day-night cycle of nearby electrical equipment, movement of inhabitants or vehicles in the environment generating noise or affecting signal paths. Periodic events detected by multiple nodes in the sensor network may result in increased traffic within a region of the network leading to congestion and possible message loss. Finally, in hostile environments, causes may include adversaries that try to compromise communication.

In this paper, we study an effective way to forecast repetitive patterns in quality of service metrics of the network, using time-series analysis. We present an application-level, autonomic routing service that adapts sensor readings routes to avoid areas that are expected to have low link-quality, while, at the same time, avoid overload of good quality paths.

We also discuss the integration of this service in the Sensor Fabric [1], a sensor networks middleware that takes care of the sensor identification, discovery, access control interoperability, data dissemination and management of sensor nodes, developed within the International Technology Alliance (ITA) project[1]. We use the extension mechanisms of Fabric to collect real-time network information on node availability, link packet drop rates and traffic loads in order to select the routes that maximise the likelihood of message delivery across the network over an unreliable multi-hop network. The routing service maintains forecasting models for each link performance metric and decides route allocation to active network paths matching node requirements. Finally, we evaluate our approach in a simulated environment and evaluate the effectiveness of network failure forecasting. In previous work [2], we have applied the forecasting model to predict node disappearance from a neighbourhood. Here, we study more extensively how forecasting can be applied on predicting periodic degradation of packet delivery rates on links and how to utilise repetitive bursty traffic patterns in order to avoid congestion due to overload of network buffers in nodes.

The rest of the paper is structured as follows: in Section II we provide background on the ITA Sensor Fabric architecture and operation. Section III presents the network performance metrics we consider in the network, the route selection method and the forecasting model we use for predicting future performance based on past observations. In section IV, we discuss integration with the ITA Sensor Fabric and the implementation of extensions for the adaptive, routing forecast service. Section V, includes evaluation of our methods using simulated scenarios. Finally, in section VI, we discuss related work from bibliography and we conclude in section VII.

II. ITA Sensor Fabric Middleware

The Fabric middleware is a network management layer that connects assets in a sensor network to clients/actors providing a publish/subscribe communication abstraction [3]. Sensors act as publishers providing data feeds based on raw or processed sensor readings. Client nodes are the consumers of this information and can subscribe to sensor feeds to receive readings as they become available. There can be multiple subscribers to published messages and publishers are not aware of the identity or address of the subscribers, i.e. there is a decoupling

[1]http://www.usukita.org/

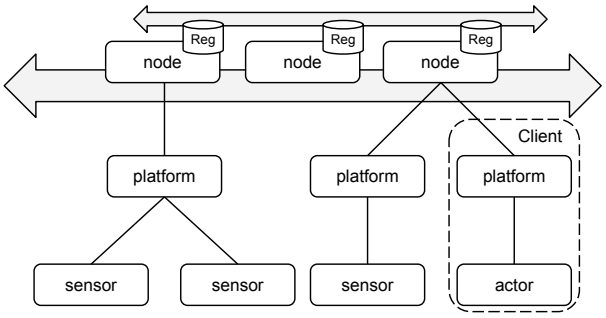


Fig. 1. Fabric component architecture

between publishers and subscribers. Clients refer to a directory service to locate potential messages types of interest that they subscribe to. The Fabric infrastructure matches publications of these messages to subscriptions and sets up routes over the multi-hop network for relaying messages to the subscriber.

Sensor networks do not, typically, form a fully connected graph, instead they rely on multi-hop end-to-end paths. Fabric supports multi-hop communication among nodes in the network while abstracting details of their location from the application developer, who perceives the existence of a fully connected network. Fabric provides the abstraction of a communication bus, where nodes can publish information, i.e. sensor feed readings, that eventually reach consumers that are subscribed to these feeds. Fabric builds an open platform of assets, where producers of information, e.g. physical or even virtual sensors, generate data that consumers, e.g. fusion centres or applications, subscribe to without imposing a single endpoint/sink in the network.

Sensor data feeds are identified using globally unique names that consumers, i.e. subscribers, can refer to and receive produced data. Information for available resources and assets, as well as real-time metrics on network status are stored in a distributed database, the Fabric Registry.

A. Fabric Components Architecture

Figure 1 presents the architectural components of Fabric. We provide a brief description of these component introduce the terminology used in the remaining paper.

Registry is a Gaian database[2] distributed among nodes in the network. It contains all information about the state of the network including node IDs and physical location, neighbouring sets, network assets, registered data subscriptions and virtual circuit paths between nodes. The database is distributed among a subset of Fabric nodes, each maintaining local data. Information retrieval happens as a query that collects data from nodes that eventually get propagated to the request point. Registry communication can take place over a secondary low-traffic link that is not subject of our mechanism as it is considered more reliable, due to the sparsity of data on it.

Node is the network endpoint of Fabric, which runs a Fabric Manager service. The Fabric Manager provides multi-hop communication and the publish-subscribe service. Fabric nodes are not to be confused with typical resource constrained sensor nodes. They have the power of a netbook computer and may have substantial battery or external power source (photoelectric cells). They run a Java virtual machine and maintain a part of the Fabric Registry that stores local runtime information. Nodes are also the extension points in Fabric as discussed later.

Platform is an adaptor that connects sensors and actors to a Fabric node. A Platform could be the equivalent of MOTE-like small, constrained device with low-power radio running on batteries. In spite of being logically a separate component to the Fabric node, a platform could also reside on the same physical device.

Sensors are attached to platforms and are the producers/publishers of information in the network. They provide feeds of data that actors can subscribe to in order to receive readings updates. A sensor may encapsulate a hardware sensing device or it can be a virtual device that produces information by consuming feeds from other network endpoints, i.e. a fusion centre.

Data Feeds are series of values produced by sensors. One sensor may provide multiple feeds, for example two separate resolution feeds from a camera or a feed with raw thermometer readings as well as their averages.

Actors are either human users or software services. Similar to sensors, they have a unique identifier that allows the middeware to route information towards them.

Client is a virtual entity, consisting of an actor and a platform through which it can interact with Fabric.

B. Extension Infrastructure

The Fabric core provides a minimum set of services required to implement a distributed communication bus service, while maintaining a small footprint and overhead in the system. Additional capabilities are introduced as plug-ins, which are grouped into families. A plug-in family is a user-defined collection of extensions that share data and management operations. Fabric allows for three types of plug-ins; *Message Plug-ins*, *Fablets* and *Services*.

1) Message Plug-Ins: Nodes process messages as they are relayed by Fabric on each hop. *Message Plug-ins* are modules that can be attached to a node's Fabric Manager to process messages directly. There are three sub-types of Message Plug-ins: node, task and actor – allowing filtering of messages that are related to any of these. Their life-cycle is managed by the Fabric Manager and they are, typically, short-lived operations, such as policy enforcement, filtering, transformation, logging, caching and encryption, without the ability to have side-effects outside their controlled environment. Plug-ins can be registered to operate either on incoming or outgoing messages of a node allowing messages to be decrypted, processed and encrypted again using different plug-ins.

Within the Fabric Manager, the *Registry* contains information about each data-feed that flows over the bus. This includes what tasks it is part of, where it was generated, who is

[2]http://www.alphaworks.ibm.com/tech/gaiandb

subscribed to it and the characteristics of its destination actors. This information is available to message plug-ins as they are applied to each individual data-feed message.

2) Fablets: Fablets are extensions that run on nodes independently of the message flow. They run in separate threads, managed by the Fabric Manager, and are more flexible than Message Plug-ins allowing a broader range of operations. They can directly access Fabric resources such as the Registry and the publish-subscribe bus, but also other non-Fabric resources such as storage devices or application databases. Typical uses of Fablets include accessing non-Fabric resources and platforms or implementation of data fusion algorithms.

3) Fabric Services: Fabric Services are the mechanism used to implement most high-level Fabric features, a modular approach that builds on Fabric's core message passing functions. Services are complementary to other plug-ins. They are separate processes that work on the side and can be attached to Fabric though the Actors mechanism to interact with the node's local bus. For instance, Fabric's sensor subscription service is implemented to provide sensor data feeds as a Service on top of Fabric's core features; communication bus, the Registry and event handling.

In section IV, we describe the family of Fabric plug-ins that have been developed for prototyping a dynamic routing mechanism for routing of sensor data avoiding links that are expected to have high message drop rates.

III. Adaptation through Forecasting

As described in the previous section, Fabric handles propagation of data from producers, i.e. sensors and fusion centres, to consumers, i.e. fusion and analysis centres via multi-hop routing over Fabric Nodes. Routing paths for the subscriptions are created on-demand, when a request for a new subscription is received or an existing one is broken. Fabric uses virtual circuit switching, as opposed to a connectionless scheme, to guarantee that packets are routed only through particular trusted nodes for security concerns. Fabric Registry contains the full catalogue of network subscriptions and their virtual circuits. In this paper, we extend the current routing mechanism of Fabric by introducing a dynamic, self-adaptive routing service that relies on forecasting link reliability and traffic patterns in the network.

A. Performance Metrics

We measure the performance and reliability of the network by collecting a set of metrics from Fabric nodes. We collect application-layer metrics for network performance that allows the approach to be independent from the underlying network. We account for node availability, drop rate of network links and traffic characteristics of feed subscriptions. Based on these attributes, we build forecasting models and periodically update multi-hop relay routes in the Fabric Registry.

A Fabric *Discovery Service* runs on nodes to track availability of directly reachable, single-hop neighbours. It should be noted that this node relationship is not necessarily symmetric as the fact that node A is directly reachable from node B does not imply that the reverse is necessarily true in a wireless network. The discovery service periodically broadcasts beacon messages to verify a node's existence to its neighbourhood. In order to conserve battery power, nodes do not constantly listen for broadcasts. Instead, they turn their radio on periodically to receive beacon messages. This process may miss some of the beacon messages, hence there is a threshold of consecutive messages that can be missed before a neighbour is considered unavailable.

Apart from availability of neighbouring nodes, the quality of the wireless links, based on measured packet drop rate (PDR), is also necessary to make a routing decision that maximises the likelihood of a message being delivered to its destination. PDR is measured by piggybacking sequence numbers on messages for each hop. Due to virtual circuit packet switching that Fabric uses, traversed nodes remain the same for each subscription, thus, per-hop sequence numbers can work. The approach has the advantage of being an inexpensive way to measure drop rates by only appending a few extra bytes on existing traffic, minimising energy overheads, however, there are some drawbacks. First, there is a non-bounded delay on metric updates. In case no messages are received by a node, either lack of traffic or large number of dropped messages can be inferred. However, the Fabric *Discovery Service* beacon message will also be affected by a link failure thus removing the node as a neighbour, which sets an upper bound on the update delay.

In case of low underlying traffic, underutilised links result in limited traffic samples weak for statistical inference. To compensate for this, additional low frequency control messages can be introduced over low-traffic links to sample their status. Furthermore, we introduce a confidence level on the link quality metric. The confidence level is a real number value in the range $[0.1, 1]$ that quantifies the statistical confidence on the observations for link PDR, based on the number of packets that have been relayed over the link. The confidence is the fraction of a minimum acceptable number of messages, c, that need to be relayed over the link in order to have a reliable metric on the link quality. We cap the confidence level to 1, even when a link accommodates more than c messages.

With regard to network traffic, nodes monitor the volume of traffic that they relay and the volume of messages they produce. Messages originating from other nodes, passing through the intermediary are counted as relayed traffic, while messages generated from a local platform attached to the node, are considered as originating traffic. Originating traffic must be sent out on the node's links, whereas relayed traffic could be rerouted to bypass the node in case of overload. They are both used to train a prediction model on future message volumes.

B. Subscription Route Selection

In this section, we describe the algorithm that selects the virtual circuits that are created for active subscriptions in the network based on the metrics described. Fabric middleware uses virtual circuit routing instead of connectionless datagrams as it targets military environments, where all nodes are not

equally trusted. Routing selection is also affected by administrator policies that are enforced by a policy management system that dictate whether some data subscriptions can only be relayed by particular trusted nodes. This would be more complex to do with connectionless datagram routing requiring per hop decisions instead of a decision at the set-up time.

We construct a link graph $G_R = (V, E_R)$ of the network, where the vertices V are the network nodes, and edges E_R are the direct links between them. Edge weights represent the expected failure rate between node pairs. Weights are calculated as a linear combination of node availability and the product of link PDR and the confidence level of the metric. The graph G_R essentially represents a map of link health in the network. Applying a shortest path algorithm on G_R between the producers and the consumers of feeds, gives a prediction for the most reliable route, i.e. the one that is less likely to drop messages in the near future.

Subscription routes in Fabric are locally cached on the nodes. When routes are updated in Fabric Registry, nodes do not immediately update their current routes. Instead, nodes update data subscription routes only when they break due to a link failure or bad reception rate that degrades below a predefined threshold. Then the producer node sets-up a new subscription path using the updated route from the Registry.

Another consideration in event-driven, multi-hop sensor networks is when an event occurs in the environment monitored by the nodes, and generates increased traffic in the network. If additional feed subscriptions are routed over the same nodes there might be increased packet loss due to congestion in node buffers. In such cases, it is preferable to separate high traffic flows to use different nodes for relaying messages. Consequently, we use the information on the traffic volumes that a sensor generates to separate high volume flows over different paths. To prevent congestion, we enhance the routing map G_R generated based on link qualities, using the expected traffic of the channels in order to prevent overloading healthy channels with too many subscriptions. To achieve this we increase link costs on graph G_R by a proportion of the overall traffic they expect to carry which penalises high traffic links. In order to determine the load of a link, we normalise the number of packets that are expected to traverse the link based on allocated feed subscriptions. All loads are expressed as a proportion of the link with maximum load. However the actual link utilisation is not known, so this could result in penalising links with low utilisation which carry a relatively high percentage of subscriptions even though the total traffic is quite low. In order to resolve this issue, the administrator can specify a threshold above which the congestion prevention algorithm would start.

Finally, the intention is to avoid routing traffic through congested links while avoiding throttling links with low to medium utilisation that can carry more traffic. Thus, instead of a linear scale on link cost penalties, we use an exponential scale so that penalisation will mostly affect the costs of highest-traffic links of the network, which are also the most likely to exhibit congestion.

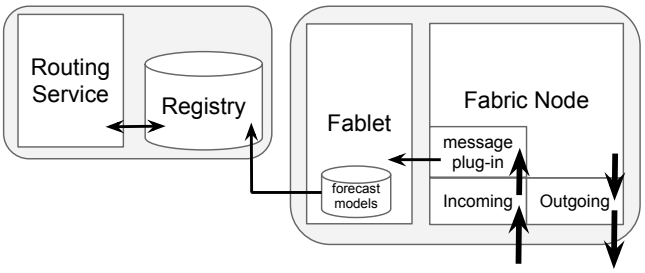


Fig. 2. Fabric plug-in architecture

IV. Integration with Fabric

A. Routing Service Architecture

In this section, we discuss the architecture of the plug-in family developed for Fabric's routing prediction mechanism and how different extensions collect metrics from the network during its operation to support the decision mechanism. Figure 2 gives an overview of the adaptive *Forecast Routing* service architecture for Fabric.

The node availability metric is already provided in Fabric Registry by the *Discovery Service*, however, we had to implement message plug-ins to measure link quality as well as generated and relayed traffic. Three message plug-ins have been prototyped for measuring link packet drop rates and message traffic load. An outgoing message plug-in at the transmitting node inserted messages sequence numbers related to a node pair, while an incoming message plug-in at the receiving node checks the sequence number to verify whether any messages have been lost from that link. The message plug-in system of Fabric permits piggyback information on messages as an extension without modifying the underlying feed subscription service. A third message plug-in monitors a node's local publish/subscribe bus for feed messages and counts them per time unit to quantify traffic of the node.

Message plug-ins are expected to be short-lived and avoid use of external resources such as hard-disk writes or network communication as this would have a performance impact on the number of messages a node can process. Thus, message plug-ins write information extracted from messages to a Fablet that is running alongside the Fabric Manager on the node. Fablets, being separate threads, have their own execution flow control and memory storage. They collect information posted by local message plug-ins and use it to update the forecasting models they maintain. The link quality and traffic load forecasting models in the Fablet periodically update the distributed Fabric Registry with new values for monitored attributes. Although the forecasting model incorporates information from all samples collected throughout a system's lifetime, it is relatively small in size – in the order of a few kilobytes. As a result, it can be serialised and stored in the Registry in a binary format. Updating forecasting models locally, rather than close to the Registry, significantly reduces the communication overhead, compared to propagating observations to a sink to perform forecasting model update outside the network.

The *Routing Service* pulls forecasting models from Fabric Registry to update its routing paths. After the forecasting phase of the algorithm, it updates the subscription routes table in the Registry used by nodes when they need to deploy new subscriptions.

B. Forecasting Mechanism

Forecasting models are produced from collected performance metrics to create two network models – the link quality and the traffic load graphs of the network projected to a future period in time. We build three forecasting models for different aspects of the network. The first model caters for recurring isolation of nodes in their neighbourhood. The second forecast model considers the packet drop rates between node links. Finally, a forecast model is used for predicting the traffic volume that a sensor feed requires.

We consider these metrics as a time-series and we use a fitting model that describes their behaviour. We have selected the Holt-Winter Additive Seasonal model that is able to capture trend as well as periodic effects in time-series. Holt-Winter applies exponentially decreasing weights on the historical data to update the model. It decomposes the time-series in three components; the level S_t, local trend b_t and the periodic factor I_t. Each of these components are updated incrementally (online) using exponential smoothing. Forecasts in the model are calculated as a linear combination of the aforementioned components as shown in equation 1, where t is the current time instance, m is the units in the future for the prediction and L is the period of the time-series.

$$F_{t+m} = S_t + b_t * m + I_{t+m-L} \tag{1}$$

We use the IBM Watson Forecasting library (WatFore) to construct and manage the forecasting models. The WatFore library provides a fully automated, extensible and scalable streaming predictive analytics framework that is suitable for monitoring any type of Key Performance Indicators (KPIs). It implements a number of streaming algorithms (including the Holt-Winters Additive Seasonal) that do not require permanent storage of historical performance measurements, thus bounding memory requirements for maintaining and using forecasting models. This is particularly important in our application domain, as sensor platforms cannot be assumed to have large storage capabilities solely for performance monitoring purposes. Furthermore, the incremental updates to the forecasting models with newly obtained measurements from continuous monitoring minimizes the processing requirements for keeping the models up to speed, imposing only marginal overhead to the sensor platform. The library also provides methods for calculating the periodicity of the performance metric using Fourier analysis, and automatic training of the forecasting models once enough data measurements have been collected.

V. Evaluation

For the evaluation of the forecasting effectiveness on route selection we emulated network scenarios that we consider fit well with expected periodic failure error classes in sensor networks. We initially evaluated the effectiveness of the algorithm for coping with node reachability and link failures, then considered congestion effects in high-traffic networks. In all scenarios, we use a grid layout, where nodes can directly communicate only with its immediate neighbours. Hence, most nodes can send messages directly to 8 neighbours while nodes at the corners are limited to 3-5 neighbours, depending on their position.

Feed subscriptions, as in the ITA Sensor Fabric framework, may originate from any point of the network. Hence, there is no single sink in the network, but there are multiple subscribers that consume data from producers. Subscribers may be terminal recipients or in turn produce new data, after processing their input feeds, which are in turn consumed by other nodes. This creates an open environment in which information does not have a single flow among nodes. We randomly generate feed subscription in the simulated network set-ups that are examined in this section.

We compare three routing approaches in the simulations. The first one is the static paths that are currently implemented in the ITA Sensor Fabric framework. This is a naïve approach that provides a lower bound of network performance – an indication of the impact of failures in the network, as it is unable to respond to them. The second approach is dynamic adaptation of routes based on the metrics discussed in the paper. However, instead of forecasting future values, route adaptations is based on recent observations. Essentially, this approach performs adaptation based on current network status. Finally, we make use of future predictions of metric values, by projecting from historic data using the Holt-Winter additive model provided by the IBM WatFore library, to dynamically adapt routes in Fabric Registry.

A. Periodic Node Communication Failures

We first study the accuracy of forecasting fail-stop communication link failures inside the network. We emulate a 5×5 network grid where 26 subscription are placed among nodes randomly. Sensor feeds produce data regularly in random intervals between 1 to $10sec$. In every run, 8 nodes, roughly $1/3$ of the population, experience periodic failures that cause them to be isolated from their neighbourhood for random time intervals. Failure times and duration are selected from the range 10 to $50sec$, with an average close to $20sec$. For this particular scenario, we assume that links between nodes are ideal and do not drop packets due to noise, in order to study only the effects of node disappearance. We emulate the scenario running Fabric on desktop machines where different nodes run in separate virtual machines and we emulate communication failures by editing linux iptables to add rules that drop packets from certain nodes in order to isolate them.

Figure 3a shows the overall packet delivery rate achieved in the network, as an average of several experiments, with three different approaches mentioned earlier; static routes (*SR*), adaptive historic routes (*HR*) and adaptive forecasting routes (*FR*). The static routing achieves a 74% packet delivery rate,

which we consider as the lower bound because there is no effort to adapt to node failures. The dynamic selection of routes based on recent historic observations improves the rate close to 85% while forecasting outperforms both, reaching a 95% packet delivery rate. Even though *HR* is able to adapt to nodes that have a longer uptime phase based on recent observation, its decisions quickly become outdated. However, *FR* by projecting these values in the future achieves better adaptation of the routing schemes as it is able to predict which nodes are going to be available in the next rounds.

As shown in figure 3a, the *FR* method exhibits, initially, similar performance with *SR*. The dive in the graph is due to the training phase that is required by the Holt-Winter model in order to start producing predictions. As soon as the model is trained and routes are adapting, a sharp increase at the delivery rate is presented. Furthermore, as the model continues to collect feedback from the network, it further improves its forecasting ability until it converges at 95% packet delivery. It should be noted that a portion of the failed messages are due to destination nodes, instead of intermediates, that have failed. In that case, there is no alternative delivery path, but the messages are still counted as undelivered.

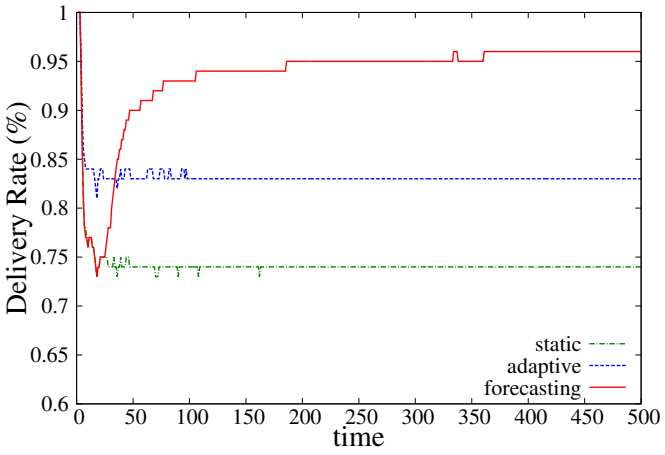
Figure 3b presents results from a similar set-up, however nodes do not have stable down/up-time periods. Instead their periods follow a Gaussian distribution with a random average value in the range of 10 to 50sec, as before, and σ value 2. Static routing is mostly unaffected from this change as it was expected. Average node downtime is not changing in the experiment, only the fixed periodicity that nodes disappear from the network. Similar situations are not uncommon in mobile networks, where patrolling nodes may occasionally come into contact with stationary nodes. Delivery rate of forecasting routing is affected by the introduced irregularity in node disappearance, though still remains high around 90%. The irregularity appears to also affect the routing based on recent observations, but not by a significant proportion (2%) to affect any change in the approach's performance.

B. Node Link Reliability

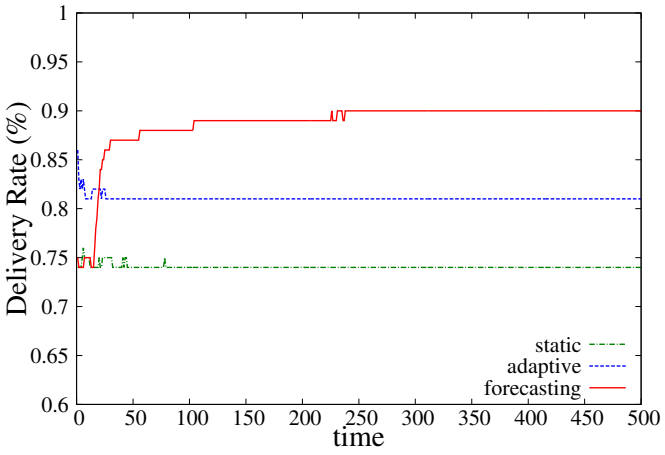
The second network aspect we study is link quality between nodes. During the lifetime of a deployed network we have noticed that links may exhibit recurring, periodic issues with delivery rates. This was typically due to moving obstacles that interfered with the signal, such as environmental inhabitants that have a certain routine, or connectivity can be affected by mobile nodes with a periodic movement pattern, even though they remain in theoretical communication range.

In order to address such repetitive adjustments on link quality, we apply the forecasting model on message drop rates of links in the network and study its effectiveness in this section. We use Castalia [4] as a simulation environment. Castalia is built on top of Omnet++[3] and provides realistic link quality behaviour in a sensor network based on traffic, signal interference, node distance and noise in the wireless medium.

[3]http://www.omnetpp.org/



(a) Periodic failures



(b) Irregular periodic failures

Fig. 3. Average message delivery rate on periodic node disappearance

To introduce the periodic fluctuation on the link quality, we modify the underlying connectivity map during the simulation. We study how feed subscription delivery rates are affected and how effective is dynamic forecasting in such situations.

Figure 4 illustrates the performance of each approach when link quality varies periodically over time. The graph presents averages for every ten rounds and the variance is illustrated as the y-axis error bars. *SR* is, again, the reference line of network degradation reaches an average message delivery rate slightly above 50%. *HR* does improve the naïve, static approach but on average it does not reach 70% message delivery rates. *FR* performs best in this case as well. After an initial training phase, of roughly 20 rounds, it increases the delivery rate slightly below 90%.

C. Traffic Load and Congestion

Exclusive use of best quality paths in the network may result in over-utilisation of nodes causing packet congestion in their network buffers. Congestion can be caused either in incoming buffers, when a node is not able to process receiving packets fast enough, or in the outgoing buffers, when the

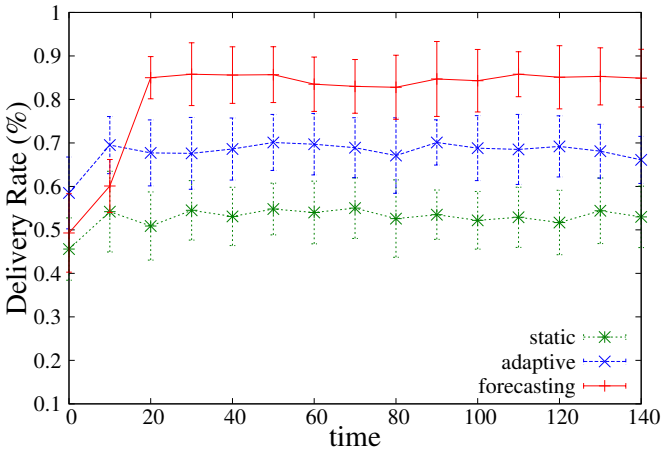
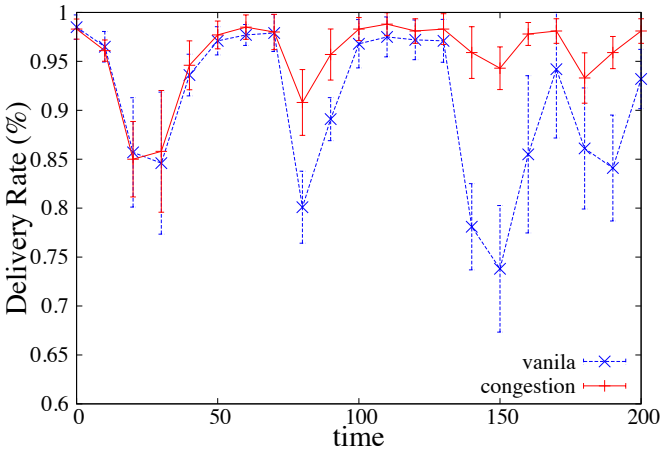
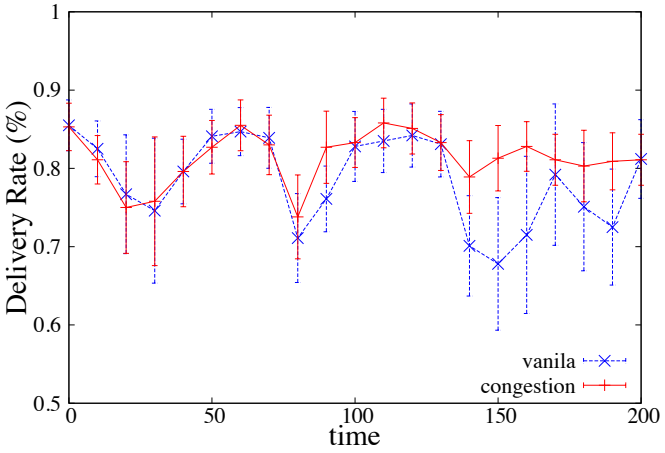


Fig. 4. Packet delivery rate over periodically unreliable links



(a) Congestion forecasting with ideal network links



(b) Congestion forecasting with realistic network links

Fig. 5. Average delivery rate on periodic node disappearance

medium is very busy for transmission and packets get queued. Castalia emulates MAC and physical layer buffers and we study the behaviour of our forecasting approach under heavy traffic. We compare the approach from previous paragraphs, which ignores traffic load on nodes, with the the traffic-aware penalisation scheme that was introduced in section III-B.

We run an experiment, where nodes generate random medium-level traffic and there are four events during the simulation that cause group of nodes in the network to generate increased traffic. Each event produces different volumes of traffic. The resulting delivery rates for this experiment are presented in figure 5. Figure 5a shows the routing behaviour in an ideal network, where links have no drop-rates apart from those in congested buffers. We ran the experiment in an ideal network in order to solely observe the impact of congestion and quantify the benefit of systematically attempting to avoid overloading nodes with excessive traffic. The four events that cause increased traffic can be easily observed in the graph as delivery rates fall sharply for the traffic-unaware scheme. On the other hand, the heavy traffic load penalisation scheme learns over time to spread traffic through different routes in the network grid avoiding packets due to congestion. It should be noted that even in an ideal network there are packet drops without heavy traffic. Such drops can be attributed to the half-duplex radio used in the simulation, where packets are lost on a node when the radio is in the transmission state.

Figure 5b presents the results of the same experiment that runs on a realistic network, where links drop packets due to noise similar to the set-up used in section V-B. Overall, packet delivery rates are lower and their variance is increased for both cases. However, the trends remain similar, where the congestion-prevention scheme performs better during high-traffic events, but the gap between the two approaches is less. This can be explained as a side-effect of the noisy links that reduce the amount of received packets, hence the effects of buffer congestion are decreased.

VI. Related Work

There are several approaches in the literature for increasing network reliability for message delivery rates, some incorporated in network management frameworks similar to Fabric. MANNA [5] and Sympathy [6] are examples of management systems that monitors nodes collecting metrics centrally for analysis. In [7] a fault management service for MANNA is described, where nodes report measurements to local managers that are subsequently propagated to a sink. Both systems make decisions on networks health based on recent observations.

Neighbourhood collaboration is utilised in [8] for detection of missing neighbours, where a protocol runs in two phases. Nodes monitor which neighbours they believe are alive in the first phase by exchanging hello messages. In the second phase the neighbourhood exchange their local observations of missing nodes and reach a local consensus before they trigger a failure alert at the sink. RedFlag [9] improves on that original algorithm, adopting some of its ideas. It requires clock synchronisation between nodes in order to begin a handshake round with their neighbours and verify their existence. If a

neighbour misses a configurable number of handshakes then a neighbourhood consensus protocol takes place between nodes on whether the node has failed. Each node tracks information about their neighbours link quality and residual energy to infer whether a failure is due to a broken link or power depletion.

The collection tree protocol (CTP) [10] is an efficient data collection protocol for multi-hop sensor networks. It is based on two main ideas for improving message delivery rates and reduce imposed overheads. A datapath validation mechanism avoids looping of messages among nodes that are formed due to dynamic link health changes and adaptive node beaconing that reduced beacon messages of nodes with healthy links to conserve energy, but increases the rate when links start losing packets. A backpressure collection protocol [11] improves delivery rates of CTP for dynamic environments with moving sinks. However, both protocols target datagram packet routing and they do not account for recurring patterns on failures and traffic.

Memento [12] is a service deployed inside the network looking for fail-stop node failures. It is based on a heart-beat mechanism that will tag a sensor failed after missing a number of consecutive heart-beats. It also introduces a variance-bound mechanism that can put an upper bound on false positives. Our approach on detecting missing neighbours is similar, as nodes periodically exchange heart-beat message to verify their proximity and they have a certain threshold of failed attempts before they consider a neighbour lost.

Regarding detection of missing packets, Silberstein et al. [13] discuss how they cope with failures in a system that suppresses updates of new values unless they exceed a predefined threshold. They compare several schemes including application level ACK messages, sequence numbers and hints of previous, possibly lost values. They, further, use a Bayesian approach at the sink to infer missing values using models learned from the data instead of interpolating.

Detection of dropped and missing packets is a concern of network protocols in most sensor dissemination protocols. Use of NACK messages has been used in PSFQ [14] and GARUDA [15] for detecting missing packets, however they require an indefinite amount of packets stored in intermediate nodes. For streaming applications, delay or lack of traffic is considered as a symptom of fault in the network [6], [16]. We chose to follow a less taxing approach of counting sequence numbers, even though the method has disadvantages that have been discussed in previous sections. However, the use of confidence factor on link quality compensates to some extent for their weaknesses.

Link quality can be measured by the ratio of undamaged received packets. Passively monitoring the link quality using snooping has been used in [17] by tracking link layer sequence numbers. Congestion levels can be monitored using buffer occupancy levels [18] or channel loading [19]. However, monitoring link quality and channel loading requires the radio to operate constantly in listening mode, thus consuming high levels of energy. Snooping has also been used in Snif [20] that operates as a secondary system with its own dedicated wireless channel, deployed on the side of to the normal sensor network for monitoring purposes.

Other forecasting approaches in the literature are focusing on predicting link availability based on node movement [21] in mobile networks. In [22] the authors attempt to introduce the lifetime expectancy prediction for nodes in the routing selection to maximise network's operating time apart from minimising packet hop-count. Furthermore, a predictive model for minimising transmission time in networks based on cross-traffic estimations has been introduced in [23]. Finally, in [24], a time-series model is proposed for predicting link quality in the network based on RSSI and LQI metrics based on a weighted average of past and present observations.

VII. Conclusion

We have presented a dynamic routing service based on forecasting network attributes, which is integrated in the ITA Sensor Fabric middleware. Forecasting trends of the network allows pro-active adaptation of routing paths for long running subscriptions, avoiding recurring network degradation. We assume the existence of a more reliable, low-traffic, secondary channel that the nodes can use to communicate with a distributed database, the Fabric Registry, in order for nodes to update network statistics for the main, high-traffic network channel carrying subscribed messages to be relayed to consumers.

We, further, demonstrated the effectiveness of forecasting of periodic failures, compared to adaptation based on recent history observations. The Holt-Winter's model used for predicting network attributes is able to distinguish different periods in the input enabling effective estimation on future node connectivity.

Acknowledgement

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] J. Wright, C. Gibson, F. Bergamaschi, K. Marcus, R. Pressley, G. Verma, and G. Whipps, "A dynamic infrastructure for interconnecting disparate isr/istar assets (the ita sensor fabric)," in *IEEE/ISIF Fusion Conference*, July 2009.

[2] T. Bourdenas, F. Bergamaschi, D. Wood, P. Zerfos, A. Swami, and M. Sloman, "Forecasting routes and self-adaptation in multi-hop wireless sensor networks," in *SPIE Defense Security and Sensing*, April 2011.

[3] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, pp. 114–131, June 2003.

[4] H. Pham, D. Pediaditakis, and A. Boulis, "From simulation to real deployments in wsn and back," *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pp. 1–6, 2007.

[5] L. B. Ruiz, J. M. Nogueira, and A. A. F. Loureiro, "Manna: A management architecture for wireless sensor networks," *IEEE Communications Magazine*, vol. 41, no. 2, pp. 116–125, 2003.

[6] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," in *ACM SenSys*, San Diego, CA, 2005.

[7] L. B. Ruiz, I. G. Siqueira, L. B. e. Oliveira, H. C. Wong, J. M. S. Nogueira, and A. A. F. Loureiro, "Fault management in eventdriven wireless sensor networks," in *MSWiM*, 2004.

[8] C. Hsin and M. Liu, "Self-monitoring of wireless sensor networks," vol. 29, 2006, pp. 462–476.

[9] I. Urteaga, K. Barnhart, and Q. Han, "Redflag a run-time, distributed, flexible, lightweight, and generic fault detection service for data-driven wireless sensor applications," in *PERCOM '09: Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1–9.

[10] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," *Proc. of the 7th ACM SenSys*, pp. 1–14, 2009.

[11] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: The backpressure collection protocol," *Proc. of the 9th ACM/IEEE IPSN*, 2010.

[12] S. Rost and H. Balakrishnan, "Memento: A Health Monitoring System for Wireless Sensor Networks," in *IEEE SECON*, Reston, VA, September 2006.

[13] A. Silberstein, G. Puggioni, A. Gelfand, K. Munagala, and J. Yang, "Suppression and failures in sensor networks: a bayesian approach," in *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 2007, pp. 842–853.

[14] C. Wan, A. Campbell, and L. Krishnamurthy, "Pump slowly fetch quickly (psfq): A reliable transport protocol for wireless sensor networks," *IEEE JSAC*, vol. 23, no. 4.

[15] S. Park, R. Vedantham, R. Sivakumar, and I. Akyildiz, "A scalable approach for reliable downstream data delivery in wireless sensor networks," in *ACM MobiHoc*, 2004.

[16] J. Staddon, D. Balfanz, and G. Durfee, "Efficient tracing of failed nodes in sensor networks," in *1st ACM international Workshop on Wireless Sensor Networks and Applications*, 2002.

[17] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *ACM SenSys*, 2003.

[18] Y. Sankarasubramaniam, O. Akan, and I. Akyildiz, "Erst: Event-to-sink reliable transport in wireless sensor networks," in *ACM MobiHoc*, 2003.

[19] C. Wan, S. Eisenman, and A. Campbell, "Coda: Congestion detection and avoidance in sensor networks," in *ACM SenSys*, 2003.

[20] M. Ringwald and K. Romer, "Snif: A comprehensive tool for passive inspection of sensor networks," 2007.

[21] S. Jiang, D. He, and J. Rao;, "A prediction-based link availability estimation for mobile ad hoc networks," *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, pp. 1745 – 1752 vol.3, 2001.

[22] M. Maleki, K. Dantu, and M. Pedram, "Lifetime prediction routing in mobile ad hoc networks," *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, vol. 2, pp. 1185–1190, 2003.

[23] S. Yin, Y. Xiong, Q. Zhang, and X. Lin, "Prediction-based routing for real time communications in wireless multi-hop networks," *QShine '06: Proceedings of the 3rd international conference on Quality of service in heterogeneous wired/wireless networks*, Aug 2006.

[24] L. Liu, Y. Fan, J. Shu, and K. Yu, "A link quality prediction mechanism for wsns based on time series model," *2010 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing*, Jan 2010.