Dynamic Routing Framework for Wireless Sensor Networks[*]

Mukundan Venkataraman and Mainak Chatterjee
University of Central Florida
U.S.A.

Kevin Kwiat
Air Force Research Laboratory
U.S.A.

Abstract

Numerous routing protocols have been proposed for wireless sensor networks. Each such protocol carries with it a set of assumptions about the traffic type that it caters to, and hence has limited interoperability. Also, most protocols are validated over workloads which only form a fraction of an actual deployment's requirement. Most real world and commercial deployments, however, would generate multiple traffic types simultaneously throughout the lifetime of the network. For example, most deployments would want all of the following to happen concurrently from the network: periodic reliable sense and disseminate, real time streams, patched updates, network reprogramming, query-response dialogs, mission critical alerts and so on. Naturally, no one routing protocol can completely cater to all of a deployments requirements.

This chapter presents a routing framework that captures the communication intent of an application by using just three bits. The traditional routing layer is replaced with a collection of routing components that can cater to various communication patterns. The framework dynamically switches routing component for every packet in question. Data structure requirements of component protocols are regularized, and core protocol features are distilled to build a highly composable collection of routing modules. This creates a framework for developing, testing, integrating, and validating protocols that are highly portable from one deployment to another. Communication patterns can be easily described to lower layer protocols using this framework. One such real world application scenario is also investigated: that of predictive maintenance (PdM). The requirements of a large scale PdM are used to generate a fairly complete and realistic traffic workload to drive an evaluation of such a framework.

1. Introduction

First generation wireless sensor networks (hereafter 'sensornets') passively transported bits from one end to another. The subjective requirements of a payload are opaque to the network

[*]Approved for Public Release; Distribution Unlimited: 88ABW-2010-5582 dated 18 October 2010.

protocols, and the role of in-network processing is limited. Various routing protocols for transporting data in sensornets have been proposed: protocols for reliable routing (2; 7; 24; 27; 28), real time communication (10; 11; 14), energy aware communication (1; 12; 29), load balanced communication, aggregation centric approaches (19) and so on to name a few. Each such protocol typically optimizes a certain set of chosen parameters in making routing decisions, and is likewise validated over a workload that only generates that type of network traffic. This means that a deployment that adopts any given protocol has to build its *entire* deployment logic using the traffic type for which the protocol is optimized. To make sensornets a viable solution to real world problems, applications need to be built on top of arbitrary communication patterns, often with conflicting requirements. For example, a meaningful case of habitat monitoring would mostly demand all of the following communication patterns to co-exist: periodic network reports using reliable sense and disseminate, critical real time alerts when anomaly is detected, aggregation to suppress duplicates, network reprogramming to transfer bulk data, patched updates for continuous customization of the sensornet, best effort communication to transfer redundant information, interactivity with the network in the form of request-reply dialogs and so on. Naturally, no *one* routing protocol can cater to such varied application requirements within a given deployment. In other words, a given deployment can be viewed as a collection of various tasks (applications) which have very different, and often conflicting, communication requirements. The deployment goal is met when the goals of its constituent applications are fulfilled.

Secondly, there is little synergy across research efforts. Pressed by scarcity of energy and a need to focus on performance, protocols are developed with little thoughts to modularity and interoperability. Though a new application deployment would have a plethora of routing protocols to choose from, these protocols cannot be readily wired together to form a communicating framework due to compatibility problems. Compatibility problems largely arise because of the assumptions made on interface and data structure requirements. In general, and as Culler *et. at.* (5) note, a framework for testing, integrating and proposing protocols is largely missing.

This chapter presents a routing framework that makes an application's communication requirements visible to the lower layers, and allows activation of application specific processing. The traditional routing layer is replaced by a highly composable collection of routing decisions. Now, the routing logic is dynamically wired as per each packets requirements. The effectiveness of this strategy is demonstrated by gathering requirements and validating a fairly complete deployment scenario: predictive maintenance (PdM) using sensornets (15).

2. Protocol Description

2.1 Routing Framework Overview

Application payload presents a three bit preamble to the framework that describes it communication intent, The framework dynamically switches routing decisions based on the preamble bits. The routing layer is now a composable set of routing components that perform similar functions, but are optimized for different classes of traffic. The routing components carry a similar three bit signature that lets the framework know their applicability for a certain class of traffic. The selection of a routing component is hence a mapping between what the application demands and what the component has to offer. The routing components house core protocol features that cater to a particular application type. This allows components to evolve independently, and owing to their composable nature, allows seamless migration from one deployment experience to another. To unify interface assumptions, the routing components share a universal neighbor lookup table that houses relevant information about various neighbors which saves storage space and makes way for consistent interface assumptions. Component protocols make ranged queries into the neighbor table to derive the best candidate hop for a given application payload. Designing such a framework requires addressing challenges of *composing* routing protocols into core components, and *regularizing* the various interface and data structure requirements. But before that, we begin with the fundamental problem of making visible an applications demands to the stack.

2.2 Specifying Communication Intent

Applications need a way to express their communication requirements in a format that is both completely expressive and minimal. Various approaches have been taken to increase application visibility to the communication framework, and much of the effort has been to prioritize data. For example, the SP architecture (20) argues for a one bit descriptor that describes the urgency of a packet. On the Internet, DiffServ uses a class of service (CoS) field, three bits in length, to specify a priority value between 0 (for best effort traffic) to 7 (real time traffic). ISP's in the present day Internet also use similar tags to differentially route packets from preferred customers and offer them a higher quality of service. However, assigning a priority for any class of traffic is a highly subjective task, and these assignment rules would not be consistent from one deployment to another. Application naming should instead revolve around fundamental communication requirements rather than blatant priorities. This would allow one to construct meaningful and consistent inferences of a packets requirement for virtually any deployment. In effect, the following question is posed: *What is the minimum number of bits to let a deployment specify its fundamental communication requirements?*

To best characterize an application to the communication framework, various possibilities exist: number of recipients (anycast, multicast, broadcast), loss tolerance, delay tolerance, priority, sensitivity to congestion or link losses, soliciting retransmissions, tagging packets for aggregation, tagging packets for load balancing, control information v/s data packets and so on. However, communication patterns can be best described using three fundamental axes: nature of payload, reliability, and time criticality.

Nature of payload: Traffic in the network can be broadly classified as data or control traffic. Data traffic is all of the push based traffic generated by a mote. Control traffic is traffic used for control plane management (like beacons, ACK's etc.), and to a certain extent, user generated traffic. Sensor networks are more than just a collection of data gathering elements that autonomously report values. There is a need to accommodate a human element into the network for a variety of reasons. Users would want interact with the network with queries, and would want to receive responses in short turn around times. More importantly, administrators see the need to continuously customize the network with updates or network reprogramming. Interplay of data and control traffic in the network needs to be closely modeled as per a deployments requirements.

Reliability: The second axis to consider is tolerance to loss. Since a deployment consists of a host of nodes that are primarily involved in data gathering, some level of redundancy is sensed values is inherent. However, dictated by application requirements, some sensed values might be loss intolerant. For example, a deployment might want to construct a time series plot of sensed values from every mote for statistical purposes. This would require every mote to reliably transfer data periodically with minimum losses. Loss intolerance is also required for network re-programming or bulk reliable transfers, where binary updates need to be transmitted to node(s) reliably over the network.

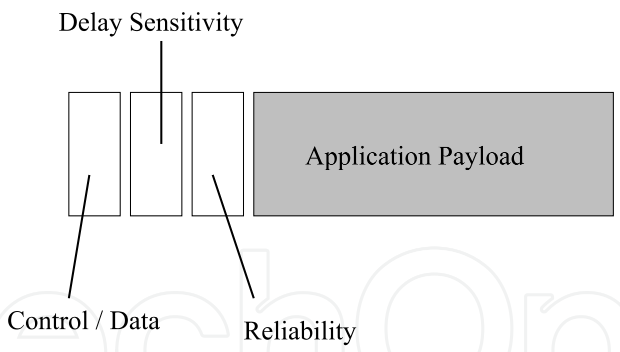

Fig. 1. 3-bit preamble appended to the application payload

Time Criticality: Some sensed value might be of no use if it does not make it to the destination within strict time bounds. Consider a deployment of sensors that report real time co-ordinates of moving objects to a camera which then pans and zooms to that area. If the co-ordinates reach the destination late, the camera might be unable to capture the desired frames of interest. Again, dictated by application requirements, there are time bounds for values of interest to make it to the destination.

2.3 Preamble Bits

Figure 1 shows the three axes of communications reduced to a simple three bit scheme that each packet carries. These bits are set or unset by the application programmer using simple API calls. The three bits, taken in combination, provide a characteristic description of an applications requirements. Figure 2 provides a complete combination of the preamble bits and their inference. For example, a beacon packet would carry a signature of [1,0,0], denoting a control packet that is loss tolerant and insensitive to delay. A data packet demanding reliability over delay would publish [0,0,1]. Similarly, a real time packet which only demands speed of delivery would publish [0,1,0]. An anomalous case is made when a packet demands both reliability *and* speed of delivery (bits [0,1,1] and [1,1,1]). Ensuring reliability inherently adds delay in transit, and such packets are interpreted as “mission critical", which see the need to both make it to the destination and in as short a time as possible. In general, the bits when combined with other information available in the packet headers make way for a powerful expression of precise communication demands. Note that the bits do not convey any notion of relative priority amongst packets, just a set of actual communication requirements.

2.4 Shared Neighbor Table: Unified view for routing protocols

Since the routing layer is now a collection of independent routing components, each component assumes the presence of various state information to be available to perform routing decisions. The neighbor table houses values such as the node-ID, energy available, congestion level, depth, link-quality estimate and a ‘last heard’ bit. The columns can easily be extended by future protocols. Since routing components share this table, it decouples core protocol features from interface assumptions and regularizes data structure requirements. This leaves the routing layer with a composable set of routing components that can be seamlessly ported across various research efforts.

Data(0)/ Control(1)	Real-Time(1)/ Non-Real-Time(0)	Reliable(1)/ Un-Reliable(0)	Inference
0	0	0	Unreliable, non Real Time, Data Packet
0	0	1	Reliable, non Real Time, Data Packet
0	1	0	Time Critical, Unreliable, Data Packet
0	1	1	Mission Critical Data packet
1	0	0	Unreliable, non Real Time, Control Packet
1	0	1	Reliable non Real Time, Control Packet
1	1	0	Real Time, Unreliable, Control Packet
1	1	1	Mission critical control packet

Fig. 2. Combinations of the preamble bits and their inferences for eight traffic types

2.4.1 Universal Beacon Packet

The wireless medium is unreliable, link qualities show time varying fluctuations, and node failures are not uncommon. Hence, there is a need to continuously monitor the state of the network by exchanging beacons at regular intervals. The various beacons assumed by component protocols are regularized by the use of a universal beacon packet. The creation and maintenance of the neighbor table is performed by the exchange of this universal beacon at periodic intervals.
The beacon packet contains information such as the ID of the node, the advertised depth, a one bit congestion indicator, and a one bit energy available indicator. A node sets the congestion bit if 75% of its buffer capacity is full. Likewise, the node sets the energy bit if less than 25% of its battery life is available.

2.4.2 Neighbor Table Creation and Maintenance

Even though there may be many potentially good neighbors available in the vicinity, there is a limit to the number of neighbors a mote can maintain due to limited resident memory . It is crucial to be able to identify the best possible neighbors and retain them in the neighbor list (27). A statistical approach to identify the goodness of a neighbor is employed.
At the start of the network, a node aggressively enters into its neighbor table every entry that advertises a depth lesser than its own. As time progresses, and owing to the presence of “gray" areas (27; 31), a node continues to hear more often from certain neighbors and less

frequently from others. Since the beacon exchange is uniform among all participating nodes, a node maintains a ratio of the number of beacons received to the number that *should* have been received since the entry was made. This ratio (p) gives a good indication of the quality of link to a neighbor by estimating the number of transmissions required as $1/p^2$ (6; 27). Link estimations apart, this ratio is helpful in establishing the true depth of a node. Consider a node which receives a beacon from a neighbor advertising a depth of n with a reception ratio of 0.3, and another neighbor with a depth $n+1$ and a reception ratio of 0.9. The given node correctly infers its true depth to be $n+2$, since it has a stronger and more stable link to the latter neighbor. In general, a node infers its true depth to be one greater than the neighbor with the largest reception ratio and least depth.
The maintenance of the neighbor table is governed by a timer driven “scan and update" (SAU) module. The module is invoked each time the timer fires, whose periodicity is equal to the beacon interval, and offset of 10 seconds. As a node continues to receive beacons from neighbors present in the table, the various fields advertised in the beacon are used to update entries in the table. Associated with every entry is a “last heard" bit which is set to true upon the reception of a beacon. SAU unsets the bit each time it is invoked. SAU also updates the reception ratio of a neighbor by using the last heard bit. An entry is evicted from a neighbor table if one of two things happen: (i) a node discovers that a neighbor′s depth is greater than its own; or, (ii) the reception ratio to that neighbor drops below 0.3 for a minimum statistical interval of 100 beacon cycles. Nodes evicted from the table are entered into a pool of blacklisted neighbors, who are not considered as potential neighbors for the next 500 beacon intervals. This prevents stale neighbors re-appearing in the table and gives an opportunity to other potential neighbors in the vicinity.
The notion of a “good" neighbor quickly blurs when there are multiple route selection components, each with their own yard of goodness measure. In general, entries are driven by their depth in the network more than by any other factor. This usually results in a good blend of neighbors with various values of depths, link qualities, congestion levels and energy values, and all of whose depths are lesser than that of the given node. Routing modules make ranged queries into the neighbor table to derive the next hop for a particular packet. However, it may so happen that a routing module fails to find any potential next hop candidate from the table. This could be either because the neighbor table is starved of good neighbors for that routing component, or due to unavailability of neighbors in the vicinity. When this happens, the SAP module is triggered with information about the routing component, and it marks for an insertion of an entry that matches the routing components needs from future beacons. This could possibly lead to an eviction of an entry from the table. The least recently used (LRU) algorithm is used to choose an entry for eviction.

2.5 Decomposing Routing Protocols to Core Components

The routing layer consists of a collection of routing components, with each component optimized for a certain class of traffic. Like the application payload, the component protocols also publish three bits that advertise their suitability for a particular application. It is crucial to decompose component protocols at the right granularity to allow rapid protocol development. As Cheng *et. al.* (3) note, choosing the granularity at which a protocol is to be decomposed is highly challenging: fine grained decomposition will result in un-necessary run time over head, while too large a granularity will fail to leverage code sharing among components and could result in significant re-implementations. They show that a composable set of of protocols that share common code result in smaller memory footprints, and are in general lucrative

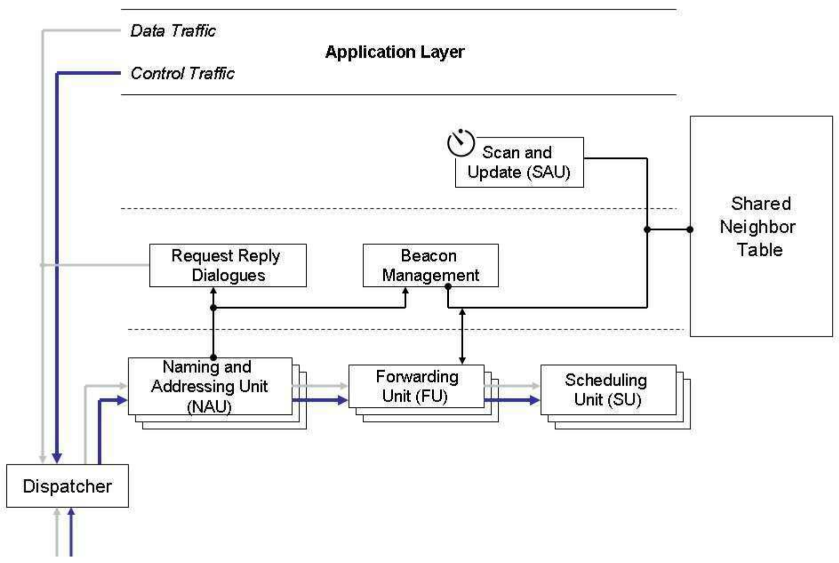

Fig. 3. Internal components of the routing protocols

considering resource scarcity on a mote. While the precise needs of future protocols are highly debatable, we decompose a routing protocol to the following components: the dispatcher, the naming and addressing unit, the forwarding unit, and the scheduling unit. The internal layout of a typical component protocol is shown in Figure 3.

Dispatcher: The first step in accepting a packet is to check the preamble bits to establish the right unit to which the packet is to be forwarded. The dispatcher is agnostic to intricacies of a protocols implementation or its naming and addressing format, and is shared by all the component protocols.

The *naming and addressing unit* (NAU) is the component that understands the addressing format for a protocol. Different addressing units are used by various protocols: while some protocols assume flat ID's, some other expect gradients or location information. Owing to the broadcast nature of the wireless medium, a node may receive a packet that it is not intended to. In effect, the NAU determines if a packet has made it to the destination, or if it needs to be forwarded further.

The *forwarding unit* (FU) is primarily concerned with establishing the next hop for a packet. Packets make it to this unit if the given node is not already the destination. The FU arrives at the next hop by making a raged query into the shared neighbor table. For example, a routing protocol that selects a route based on link qualities, congestion and depth at its neighbors would make a ranged query of the form $f(link\_quality, congestion, depth)$ into the neighbor table to arrive at a list of potential neighbors. The FU then applies its set of optimizations to arrive at the best neighbor(s) to which the packet is to be forwarded.

The *scheduling unit* (SU) is a buffer data structure that can order packets awaiting transmission. It can internally schedule the order in which packets are to be forwarded to the link layer for injection into the wireless medium. Certain protocols might need to buffer packets for potential retransmission, while other might need to hold packets to perform aggregation stalling for similar information to arrive. Certain other protocols might want to reschedule the order of injecting packets into the link layer to transmit mission critical alerts ahead of regular traffic.

Apart from these units, there are certain other units in the routing layer that accept packets for further processing. For example, beacon packets make it to insertion/eviction module, which considers the beacon as a potential neighbor. Likewise, control traffic originating from the base station is a typical query into the node, to which the node responds by performing certain local computations. The shared neighbor table is controlled by the scan and update unit, which periodically scans through the neighbor table to evict stale neighbors and enter them into a blacklisted pool.

2.6 The Dynamic Routing Framework

The effectiveness of using three bits to drive customization and protocol selection within the dynamic routing framework is next investigated. Apart from communication requirements of reliability or delay tolerance, the bits fundamentally divide the traffic as being of control or data type. Control or data traffic do not necessarily demand differential routing in terms of shorter or longer paths. In fact, most of control traffic (like beacons or ACKs) are for one hop use only. Beacons in particular are simply broadcast, and require no route selection or optimization. Communication requirements for control and data packets are best characterized by a need to be *scheduled* differentially inside the framework, and then, routed optimally. Differential treatment is achieved by the use of separate virtual queues for control and data traffic, while routing components are dynamically switched based on demands for reliability and delay. A detailed interaction diagram of packet with the framework is shown in Figure 4. Application presents a blend of control (C0-C3) and data packets (D0-D3). The suffix indicates the status of the reliability and real time bits. For example, packet D3 would have preamble bits set to [0,1,1], with the first bit indicating a data packet and last two bits account for the suffix 3. Selection of a routing component is driven by the suffix number. In other words, both Dx and Cx are offered the same routing component. Data or control traffic, however, are scheduled differently after route selection is established. Two virtual queues, one each for data and control traffic, take the incoming packets and schedule them for transmission to the lower layers of the stack.

2.6.1 Routing Protocols

An implementation of the component routing protocols in the framework is next discussed. Classifying traffic on reliability and delay leads to four combinations of routing components. A possible route selection strategy for four classes of traffic are described as follows:

Reliable, non Real Time Routing: Numerous approaches have been taken for this class of routing in the past (7; 24; 27; 28), to name a few. These routing protocols emphasize reliable delivery over time to deliver. A maximum of three link level retransmissions are used to ensure successful reception. Since a retransmission is costly in terms of energy and bandwidth, the protocol selects the strongest link to a neighbor closer to the destination. High estimations of link quality, or a form of ETX (7), is the most lucrative option here. The packet takes numerous short hops of high quality links to make it to the destination with minimum losses. Hence, speed is compromised by the use of numerous short and reliable hops. Delay tolerance for this class of traffic make it highly conducive to aggregation, since the SU can stall for self-similar data to arrive. Nodes with growing congestion and low energy are avoided as much as possible as the next hop.

Unreliable, Real Time Routing: Certain classes of traffic have a time associated with them, after which the data is virtually of little or no use. Speed of delivery is emphasized, and losses can be tolerated to a certain degree. Various protocols have been proposed for this class of

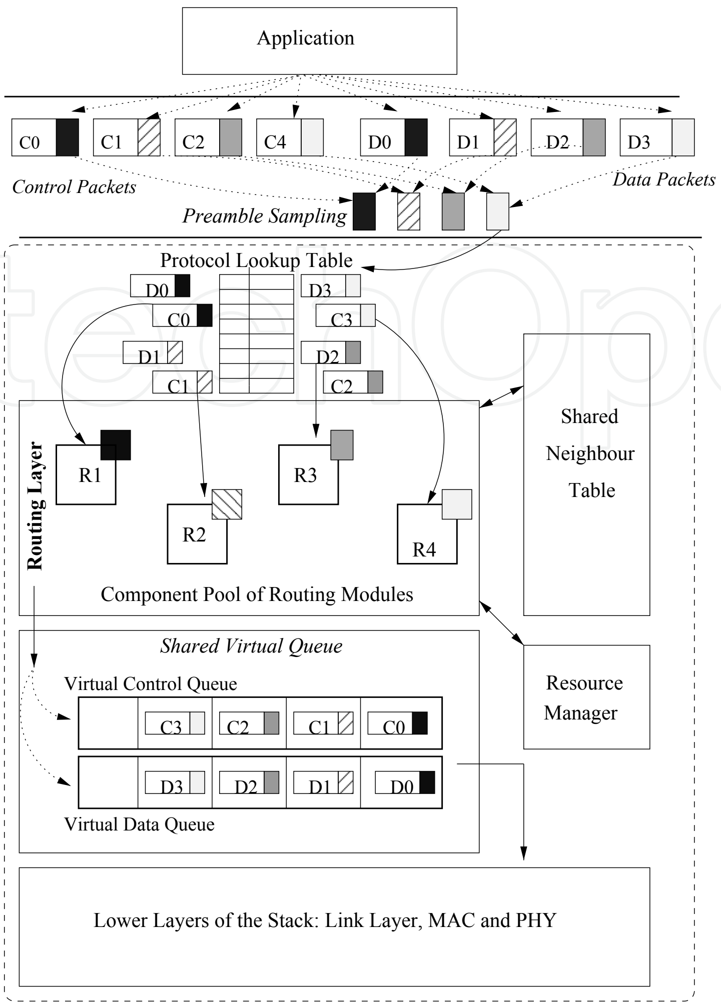

Fig. 4. Overview of the dynamic routing framework

traffic (4; 11; 23). This approach greedily forwards traffic to the base station primarily based on depth, avoiding congestion and low energy nodes, and ignoring link quality estimates or indications of poor packet reception.

Reliable, Real Time Routing: A contradictory requirement where packets need to maximize their chances of reception and yet want the shortest path to a destination. Waiting time for ACKs and retransmissions only add to the delay of the packet in transit. To both meet delivery deadlines and thwart link losses, the routing protocol first chooses nodes with the minimum depth. Of this list, nodes with the highest link quality are chosen, and the protocol injects *multiple* copies of the same packet to minimize link losses. The exact number of duplicates is based on the estimated link quality of a particular node. For example, if a neighbor with a link quality of p is chosen, the protocol transmits $\lceil 1/p \rceil$ packets with the same data. This form of routing is costly in terms of resources, but this type of traffic is reserved for mission critical alerts which is arguably rare in the network.

Unreliable, non Real Time Routing: This form of routing is applied to packets that neither demand reliable transmission, nor demand a speedy delivery. A good fraction of sense and disseminate traffic would fall into this category, where data values exhibit significant redundancy. Every packet adds little value to already existing information in the network, perhaps even repeating known observations. This form of traffic is also predominant in delay tolerant networks. This form of traffic is an excellent candidate for aggregation, where the SU unit can stall for long windows of time to effectively aggregate observed results to reduce the traffic to a handful of packets. This form of routing makes a compromise on both link quality and depth to route packets to the destination. Alternatively, this form of data centric routing could also utilize a minimum Steiner tree to maximize chances of aggregation.

2.6.2 Virtual Queues

Packets that egress the routing modules would need to be passed onto the link layer queue for subsequent transmission. Scheduling packets into the link layer queue is controlled by two virtual queues, one each for data and control traffic. The virtual queues do not maintain any independent buffer of their own: they only maintain the order in which packets shall be forwarded to the link layer. While packets await their transmission, they are physically housed in the scheduling unit of their respective component protocols. The scheduling unit internally marks a packet to be ready for transmission after it performs its set of optimizations (e.g., aggregation, reordering etc.). Fair scheduling is employed amongst the four scheduling units into the virtual queue, and from the virtual queues themselves to the link layer. This ensures that an even mix of control and data traffic are output from the framework, even though one or the other type may have a *different* packet arrival rate. In effect, this schedules both traffic types evenly by policing their resource sharing.

3. Related Research

The idea of classifying network traffic and offering differential services is not new to the networking community. The Internet witnessed the proposal of the DiffServ mechanism. DiffServ assigned integer numbers to derive Class of Service (CoS). The CoS varied between 0 and 7, and three bits were used to specify a value in that range. Each router would then apply a differential treatment to each class of traffic. Likewise, the two bit differentiated service architecture (17) is also in line with our approach. However, DiffServ in general required intelligence embedded in the routers, something which contradicted Internet's philosophy of end system intelligence. It was virtually impractical to maintain behavioral consistency amongst the millions of routers deployed in the Internet. Also, with the advent of optical interconnection lines, link errors were reduced to a bare minimum and the network capacity in terms of bandwidth shot up. The Internet could very well handle most of the demands placed on it in terms of voice, video and data traffic placed on it without the need for DiffServ. It was hence pondered that the DiffServ was a "technical solution to a technical problem that did not exist". The conditions in sensornets, however, are highly fertile for a differentiated service mechanism. Links are error prone, bandwidth is scarce, energy conservation is of utmost importance, and hop-by-hop processing and intelligence are the key to building networking technologies. Service differentiation is beyond a mere luxurious add-on in sensornets, it is more of a basic requirement. Meeting the subjective requirements of various traffic classes is a key to building successful deployments.

On the sensornet side, there has been an excellent series of work that address the problem of promoting synergy and providing sufficient abstraction for rapid protocol development. Work by Culler *et. al.* (5) argued that the abstraction that IP provides in the Internet can be provided at the link layer for sensornets, and this was substantiated by the SP architecture (20). The case for a modular network layer, where core network layer protocols were decomposed into an abstract set of modules, was then proposed by Cheng *et. al.* (3). Dunkels *et. al.* (8) proposed an architecture that could adapt to heterogeneous protocols. The idea of building a dynamic routing framework also aligns with the ability to build dynamic networking stacks (18) with a major focus on performance and efficiency, which was proposed to be built on top of the x-kernel (13). However, none of the above architectures raise the issue of conflicting application requirements, and the presence of such a collection of applications in real world deployments. A logical extension of the above efforts is to address ways to specify precise communication patterns for every application, and building a dynamic protocol framework that can cater to such requirements.

The issue of supporting concurrent applications, with each independent application sharing the resources of a mote, has been addressed in Mate (16) and Melete (30). Mate is primarily concerned with code dissemination for network reprogramming. Melete significantly extends the Mate architecture, and addresses the issues of reliable storage and runtime sharing of multiple concurrent applications. Though it effectively substantiates the possibility of running concurrent applications on a mote, it does not address the conflicting communication requirements of various applications. Our work complements their by utilizing that fact that multiple concurrent applications can reside on a mote, and we address every application's communication requirements.

As far as routing protocols themselves are concerned, sensornets are at a stage where sufficient exploratory work has been performed. As each designer explored a potential use of sensornets, new protocols began to emerge. However, the journey has been fraught with severe challenges. As Fonseca *et. al.* (9) note, the trivial case of flooding and tree construction on motes took "three years and five successive implementations". We are not aware of dynamic and composable protocols that can cater to a wide range of application demands in sensornets. However, in our quest to build such a framework, we leverage the vast body of work available in sensornets.

4. Application Background

With the dynamic routing framework in place, the various intricacies of a fairly complete large scale real world deployment of Predictive Maintenance (PdM) is analyzed as an example target application. We begin by gathering the various communication demands of this deployment. The three bit scheme is employed in exposing these requirements to the communication framework. Using these requirements, we chart a comprehensive traffic workload to drive an evaluation of the dynamic routing framework.

4.1 Predictive Maintenance

Productivity is a key weapon for manufacturing companies to stay competitive in a growing global market. Increased productivity comes through increased availability. The need to be continuously available has driven many companies to focus on effective maintenance strategies. PdM is a set of techniques which help determine the condition of in-service equipment in order to predict when maintenance should be performed. PdM uses a variety of vibration analysis, oil analysis, infrared thermography and ultrasonic detection with an objective of reducing catastrophic equipment failures, and the associated repair and replacement costs much before the failure occurs. PdM enables machinery stakeholders to monitor, access, predict and in general understand the working of physical assets.

One alternate way of implementing PdM is by use of online motoring systems of wired sensors that can continuously gather statistics. Wired sensors are constrained by a 1:1 point-to-point connectivity links, and such systems are expensive to procure and install. This trend in

general has limited their penetration into the market. In fact, most small to medium sized companies would not procure them owing to their higher cost and lower return for investment, resulting in their mere 10% market penetration (15). Sensornets break ground by promising to be cheaper, providing higher returns on investment, flexibility owing to a broadcast wireless communication pattern, self-healing abilities when it comes to node failures and adapting to changing network conditions. Hence, there is great potential for sensornets to tap into the market provided they can completely satisfy all of a deployers requirements by allowing them a greater flexibility in communication patterns.
The case for PdM as a potential killer application for sensornets has been investigated by Krishnamurthy *et. al.* (15). They create an excellent groundwork for PdM using sensornets by demonstrating a viable return on investment. However, their focus is primarily on the effect of hardware architectural impact on performance. Their study is limited to PdM vibration analysis, where the deployment generates just one type of traffic: vibration signatures that need to be reliably transported to a base station. We significantly extend PdM's requirements in terms of complete *communication* requirements from a deployers point of view by enumerating various communication patterns possible. Accommodating the various communication patterns in such deployments is an important problem to solve. This is because the cost of deploying and running such an infrastructure would be dominated by *software and service management*, while diminishing hardware costs would make it cheap to network hundreds of motes. In effect, our study builds upon and complements their work, and we show that sensornets can be a powerful platform to perform PdM.

4.2 Requirements

Most deployers want a system of sensors to autonomously report sensed values and raise alerts, while they also want the presence of a human element in the process. Human intervention can be best characterized by an ability to interact with the deployment (query-response), apply patched code updates, re-program the network with newer protocols, manage or change various threshold values and so on. In general, there is a constant need to evolve, interact and update the deployment with changing requirements, which allows for continuous customization of the sensornet behavior.
Based on our survey of PdM requirements, we have found that deployers wanting to adopt sensornets as a means to build their maintenance regime demand the following aspects:
Req. 1. *Periodic reports*: Deployers expect the network to report sensed values on a regular interval. Such statistics make way for monitoring, analysis, trend forecasting and prediction of equipment behavior which can help them chart maintenance schedules. Apart from maintenance predictions, it also helps assess performance of new machinery within a warranty period.
Req. 2. *Streaming Real Time Values*: Apart from periodic updates of sensed values, there are numerous instances when there is a need to report values in *real time*. Examples could include protected zones with cameras which need to capture a video of a personnel when he walks into the facility. This would require sensors to report co-ordinates in real time to camera installations, which would then pan and zoom to capture streaming video.
Req. 3. *Query-Response*: Users or administrators need to query and get a response from the network at will. As with any interactive system, there is a need to maintain short turn-around times within user irritation levels. In fact, there are various levels of accuracy and delay associated with each response. Some responses need to be time critical, while others demand accuracy with more acceptable levels of turn around times. In general, while keeping turn

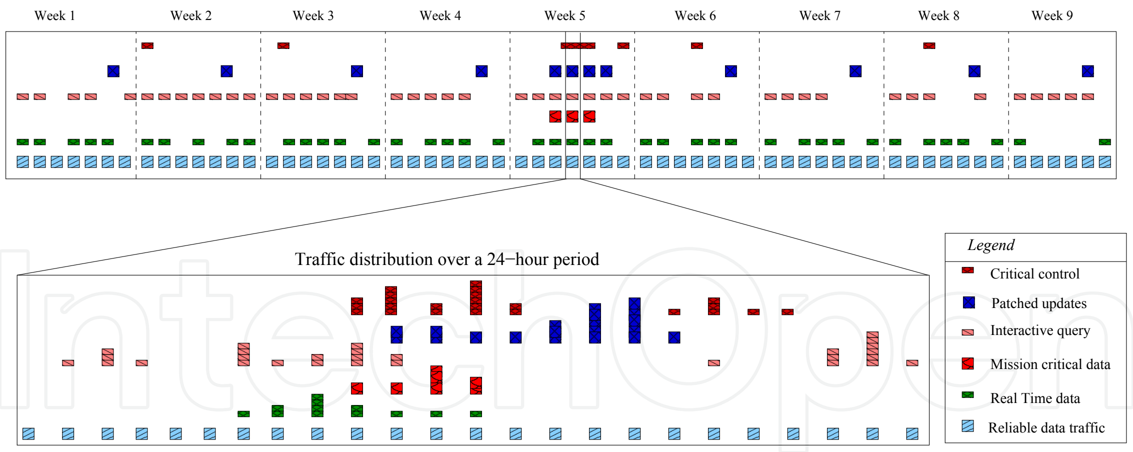

Fig. 5. Traffic generation for a 9-week PdM derived from requirements. The workload generates periodic samples, real time streams and mission critical alerts. The user/administrator interacts with the deployment at will with queries, weekly patched updates, and one major network reprogramming.

around times low, there is a need to model such subtle variations even within these interactive dialogs. Deployers further stress on the need to keep high levels of interactivity at all stages of the deployment, especially when mission critical events occur.
Req. 4. *Continuous Customization*: Requirements are not static, and would continuously evolve with time. This means that administrators see a need to constantly update or change network behavior. Examples could include changing threshold values for alerts, commanding specific motes to raise or lower sensing fidelity and so on. This in general emphasizes robust communication between an administrator and the network, and could even involve minor and incremental changes to network software.
Req. 5. *Network Reprogramming*: Change is inevitable, and must so with network protocols. Deployers want the liberty to completely reprogram an entire sensornet deployment when a more stable or efficient communication suite is available. Such modes of communication emphasizes on the need to reliably bulk transfer large pieces of code to the entire deployment.
Req. 6. *Mission Critical Alerts*: Apart from the above modes of communication, there are certain mission critical events which need to be reported reliably in as short a time as possible. Such events may trigger a cascade of events within the network. In fact, there is a need to accommodate a large number of interactive queries and commands to various motes when such events occur. In short, when serious anomaly occurs in the deployment, there is a need to gracefully alert, shutdown and recover as much as possible before the situation exacerbates.
These set of requirements are not isolated to the case of PdM alone. Virtually any practical sensornet deployment, if put under a microscope, would usually result in an enumeration of similar requirements to take place *concurrently* throughout the lifetime of a network with varying levels of emphasis on one requirement or another.
The requirements reveal significant variations in terms of communication needs. Variations exist in terms of reliability, time criticality, mission critical alerts and levels of interactivity with the deployment. We proceed to show how these set of requirements can be made visible to our framework using just three intent bits.

4.3 Translating Requirements using Three Bits

We show how just three bits can be used to make visible all of PdM's requirements to the communication framework. We proceed by identifying the nature of traffic (data/control) from the requirements, and subsequently analyze requirements of reliability and delay in transit.
The first step is to distill control from data traffic. We broadly define data traffic to be traffic that a mote creates using a sensed value, for consumption at its destination. Control traffic is all of those packets that are used for control plane management (e.g., beacons), and traffic from end users. User generated traffic includes queries, responses, commands, and code updates. This is done to ensure that user traffic, along with control plane traffic, is scheduled differently from sensed data. While traffic is normally low during normal operations, critical events in the facility are usually coupled with a surge of traffic in the network. It is precisely at such times that congestion begins to surface (26). And again, at such times, it is highly likely that an end user or an administrator will issue queries, commands or updates into the network. In effect, while the data plane reports values of interest, user interactivity and network management at such times are not compromised. For the rest of the discussion on mapping various requirements using three bits, refer to Figure 2.
Data traffic can be easily grouped as per their requirements of reliability and speed of delivery. PdM requires periodic reports from every sensor to build a statistical record of the facility (Req. 1). This type of traffic publishes [0,0,1]: data traffic demanding high reliability and no urgency of delivery. Real time streams (Req. 2) would publish [0,1,0], stressing on urgency and lesser demands of reliability from the network, since the payload could be rendered useless if reception is delayed. Mission critical traffic (Req. 6) would publish [0,1,1], emphasizing on both urgency and reception.
Beacon packets publish [1,0,0]. Since they are usually broadcast into the medium, there is no notion of reliability or time to deliver. Interaction with the network (Req. 3), however, can be modeled at very fine granularities with these three bits. Using the axes of reliability and urgency, four levels of interaction are made possible. For example, queries of type [1,1,0] would result in very fast turn around times, but with compromised accuracy. Interactions using [1,0,1] preamble would be high on accuracy, but with longer turn around times. Critical real time alerts or updates shall present [1,1,1], which would ensure short turn around times *and* high accuracy. Packets for patched updates, commands or network reprogramming (Reqs 4 and 5) would publish [1,0,1], which stress on reliable delivery. To summarize, we tabulate the various requirements mapped as one or other traffic type in Figure 6.
We conclude our mapping by summarizing the criterions to validate PdM's requirements: (i) Types 1 and 5 should have high delivery ratios compared to other forms of traffic, with an acceptable compromise on transit delay; (ii) Types 2 and 6 should have short transit times, with a compromise on reliability; (ii) Types 3 and 7 traffic should have high delivery ratios *and* short transit time; (iv) interactivity with the deployment should be maintained at all time, even in times of growing congestion and mission critical events. This in general emphasizes differential treatment to data and control traffic.

4.4 A Realistic Traffic Workload

We use PdM's requirements to create a realistic traffic generation scenario of a typical operation to drive our evaluation. We are interested in fairly large scale operations, and seek to identify the various activities that shall typically take place over a large window of time. We reword the requirements in terms of traffic generation, and create a comprehensive workload to drive our evaluation.

All traffic from the motes converge towards the base station. The base station issues queries to random motes, and replies are likewise routed back to the base station. The base station applies patched updates to specific motes, while network reprogramming is applied to every mote. Every mote generates a sensed value at a periodicity of one hour (Type 1), which needs to be reliably transported to the base station. This would help the deployment build a statistical monitoring record of the facility to drive maintenance. Nodes also generate asynchronous real time streams of value, 5 packets at a time (Type 2). Real time streams are generated at a mote with a probability of 0.1 at any instant of time, which needs to be transported to the base station as fast as possible. End users generate queries at will modeled as a Poisson process. Each query has a random flavor to it (Type 4-7), some queries demand high interactivity (small turn around times), while other demand high accuracy (commands, queries or patched updates). The administrator performs a network wide patched update roughly once a week, when the network is fine tuned to slightly update or customize behavior (Type 5). Nodes observe mission critical data with a probability of 0.05, and generate 5 packets at a time (Type 3). Each time this happens, it triggers a surge of real time streams of values at motes close to the phenomena. The situation is responded to by the user, who generates multiple queries to the zone, and with a probability, issues numerous commands and patched updates.
We use this pattern to generate synthetic workloads for a 9-week evaluation period of a large scale plant employing PdM. The generation is better captured in Figure 5. Unless otherwise stated, we consistently use this traffic workload for every experiment.

5. Simulation Results

We evaluate our dynamic routing framework by both simulations and experiment on testbeds of MicaZ motes. We resort to a simulation based study to analyze behavior at scale where we simulate behavior for thousands of motes. Though a simulator tends to hide certain physical characteristics of actual motes, it allows us to demonstrate the effectiveness of our framework for operations at various scales and densities. Nevertheless, we make every effort to model network behavior to as close to reality as possible. Link behavior between any pair of nodes is closely modeled around results by Woo (27) and Zhao (31). Using that model, we also implement the typical broadcast behavior of all packet transmissions and receptions that take place in the network. As our next round of results, we also implement our framework and evaluate it on a 40-node MicaZ testbed.

5.1 Evaluation Criteria

Evaluation is driven by a comprehensive 9-week long workload detailed in Section 4.4. The workload generates a good blend of the various traffic, while carrying out the requirements of PdM. Our first set of results analyze behavior of a dynamic routing framework at scale. We vary the number of nodes from 4 to 1024, and show how PdM's subjective requirements for various traffic types are met at scale. In effect, we show that our framework can adapt to the demands of: (i) reliability for traffic Types 1 and 5; (ii) the real time nature of Types 2 and 6; (iii) the mission critical nature of Types 3 and 7, which demand both reliability *and* short turn-around times; and, (iv) interactivity with the networks using Types 5, 6, and 7[1]. We show that such diverse communication patterns can co-exist, and also meet their subjective

[1] We do not generate Type 0 traffic, since none of PdM's requirements map to this. Also, we do not profile Type 4 since it is made up of beacon and ACK packets in our workload.

Preamble bits	Inference	PdM Req	Traffic Type
[0,0,0]	Unreliable, non real time packet; Exhibits significant redundancy	*X*	Type 0
[0,0,1]	Reliable data traffic, demands high throughput; agnostic to delay	Req. 1	Type 1
[0,1,0]	Real time data stream, demands short delivery time; agnostic to loss	Req. 2	Type 2
[0,1,1]	Mission critical data, demands reliability *and* speedy delivery	Req. 6	Type 3
[1,0,0]	Unreliable, non real time control packet;	Beacons, ACKs	Type 4
[1,0,1]	Reliable control traffic; agnostic to loss	Req. 3, 4, 5	Type 5
[1,1,0]	Real time control packet; demands speedy delivery	Req. 3	Type 6
[1,1,1]	Mission critical control; demands reliability *and* speedy delivery	Req. 6	Type 7

Fig. 6. PdM's requirements mapped to various traffic types

demands. Since the workload used mimics PdM operations, our results in turn validate the deployments goal for operations at scale.

5.2 Delivery Ratio

We begin by analyzing the average end to end success rate for all the traffic generated by the application (Figure 7).

We see good delivery ratios for reliable data traffic (Type 1), reliable control traffic (Type 5) and mission critical control information (Type 7). High delivery ratio for Type 1 traffic validates PdM's expectation of samples from every mote at periodic intervals. Type 5 traffic denotes interactivity with demands on accuracy, which are likewise met. Reliable traffic in general is driven by a retransmission upon failure, which significantly raises chances of successful packet reception. Similarly, the odds that mission critical traffic (Type 7) makes it to the destination are also high. Mission critical control traffic would be routed through the control queue, and it maintains a high delivery ratio of around 0.9. Delivery ratio is poor for real time data traffic (Type 2), which is routed greedily based on a shortest path algorithm.

While similar routing is applied to both Type 1 and Type 5 traffic, the delivery ratio of reliable data (Type 1) dips a little with increasing number of nodes as compared to Type 5. This is mostly because data traffic is mapped onto a separate virtual queue than control traffic. While Type 1 would face rising congestion with increasing number of nodes, Type 5 traffic hardly witnesses any of this. Likewise, delivery ratio for real time queries (Type 6) are significantly higher than real time data streams (Type 2). Mapping data and control traffic in separate virtual queues has enabled us to provide high levels of interactivity with the network.

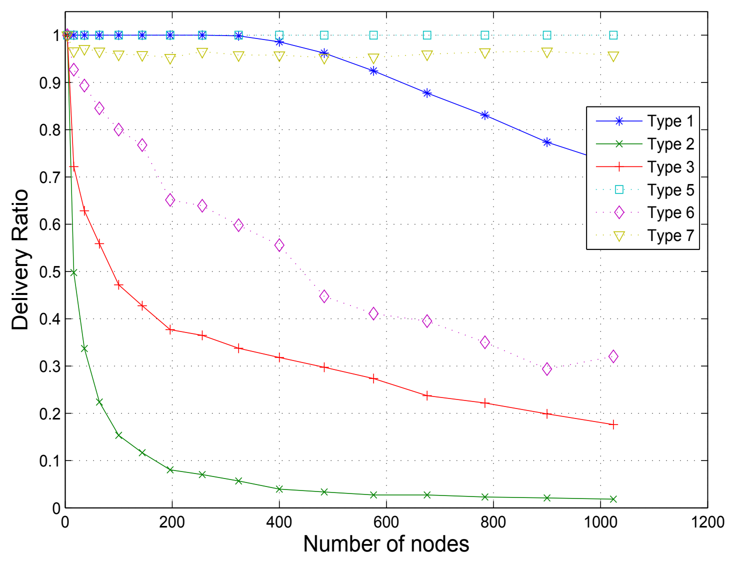

Fig. 7. Delivery ratio for different traffic types

5.3 End to end delay

We profile the average end to end delay experienced for all traffic that makes it to the destination. Delay in the network is primarily a function of the number of hops a packet takes, and the queuing delay at every hop. As a packet increases its hop count to destination, end to end delay intuitively increases.

The average end-to-end delay for all traffic types for increasing number of nodes is shown in Figure 8. All traffic demanding speedy delivery experience short transit time. Real time data experiences least delay (Types 2 and 6), reliable traffic experience maximum delay (Types 1 and 5), while mission critical traffic experiences delay that is only marginally more than real time data.

Small delay profiles for Types 2 and 6 directly validate PdM's requirements of real time streaming communication. Likewise, short delivery times of mission critical traffic (Types 3 and 7), coupled with their high delivery ratios, validates PdM's requirements for mission critical communication. High delay values for Types 1 and 5 only emphasizes on their ability to select numerous short hops of high quality.

The interplay of control and data traffic, which receive differential scheduling due to separate virtual queues, is also captured in the plot. Overall, data traffic experiences more delay than control traffic. For example, though Types 1 and 5 are offered the same routing, the overall delay for Type 1 is almost twice that of Type 5. This highlights the ability of the framework in meeting PdM's overall requirements of maintaining network interactivity. With PdM's overall objectives met, we now analyze the causes of losses in the network.

5.4 Link Losses

Effectiveness of the various routing components in their ability to choose the right path for every traffic type is best characterized by profiling the link loss distribution. Link loss is a typically attributed to the unreliable wireless medium, where packets are corrupt or lost while in transit. Routing components that stress on reliability need to understand the nature

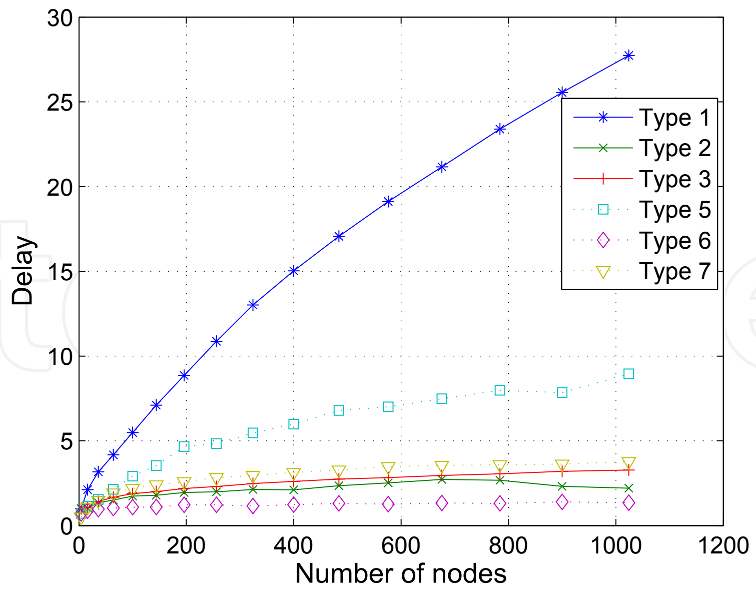

Fig. 8. End to end delay for different traffic types

of links, and use these to derive a suitable next hop while keeping the requirements of the payload consistent.
We profile link losses for various traffic types in Figure 9. As the number of nodes in the network increases, so does the effective number of hops that a packet takes to reach its destination. This in effect increases the probability of a link loss. Real time data streams (Type 2) experience maximum link losses, largely because of the nature of route selection which greedily forwards traffic to nodes closest to the base station. Reliable traffic (Types 1, 5), however, make ranged queries into the neighbor table with high thresholds of link estimates. Likewise, they experience nearly zero link related losses in the network. Because of inter-node spacing in this experiment (10 feet), neighbors closest to a node do not fall over into the gray area. Mission critical alerts (Type 7), likewise experience low values of link losses since they thwart link error by multiple copies per packet transmission.

5.5 Congestion losses

Congestion occurs when nodes inject more packets than the network can handle. While our workload generates traffic that can normally be serviced by the network, congestion does occur for a variety of reason. First, all data traffic is destined to one node (base station). Hence, all of the network's traffic converges towards nodes closer to the base station to be routed via them. Even though we try to avoid congested nodes in route selection, a point comes when all neighboring options for a node are congested. Congestion particularly increases with rising number of nodes in the network, which simply translates to rising traffic levels for nodes near the base station to service. Based on PdM's requirements, we also notice that congestion is likely to occur when serious anomaly is detected. When a mission critical failure is noticed, a surge of events takes place in the network. Nodes report mission critical alerts, and some other nodes in the vicinity would begin to send streams of real time values. The end user or administrator would add on to this by issues commands, queries and triggering actions. In our workload, both these causes are sufficiently represented. We now analyze the

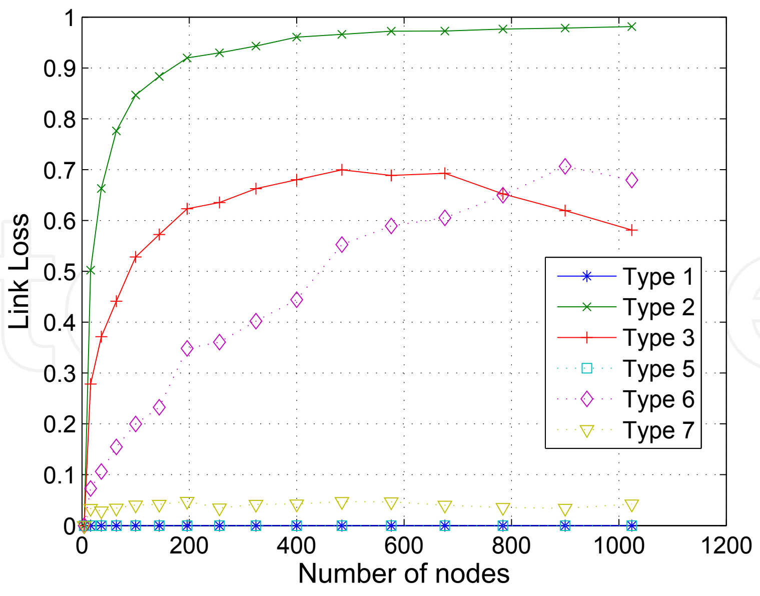

Fig. 9. Fraction of packets loss due to link losses

role congestion plays in the network, and profile the various congestion related losses for the traffic types.
The fraction of packets lost due to congestion are shown in Figure 10. For network scales of a few hundreds of nodes, congestion is not really a pressing problem because of the low duty cycle of nodes. However, congestion starts to surface for networks with more than 300 nodes, primarily because of increased load on nodes closer to base station. We notice that Type 1 traffic witnesses maximum congestion related losses. As packets begin to approach the base station, traffic from other types (real time streams or mission critical alerts) would try to avoid congested nodes nearby and choose low quality links with faster transit times. At this same stage, reliable traffic would take two or three additional hops to ensure high quality links.
It is interesting to see that mission critical data (Type 3) also experiences congestion losses. This has a few implications for congestion control in general. When mission critical anomaly is detected, activity of motes suddenly peaks. Various nodes start to simultaneously inject traffic into the network. Congested links, coupled with multiple copies per packet from Type 3, only makes matters worse for mission critical data. This suggests that dropping *any* packet in a FIFO manner, as most current congestion control schemes do, only undermines performance. In general, utilizing information about nature of payload and dropping packets of relatively lesser importance should be an added metric to future congestion control algorithms. Lastly, we also observe that control traffic (Types 5, 6, 7) do not experience congestion drops. This means that even in times of congestion, interactivity is kept high because control traffic is offered differential scheduling. This further validates PdM's requirements of maintaining high interactivity with the network even in times of congestion and mission critical events.

5.6 Interactivity with deployment

While the effects of scheduling control and data traffic differentially are brought out, we seek to understand the interplay of various types of interactive control traffic within the virtual 'control' queue. Three levels of interactivity are made possible by the use of preamble bits:

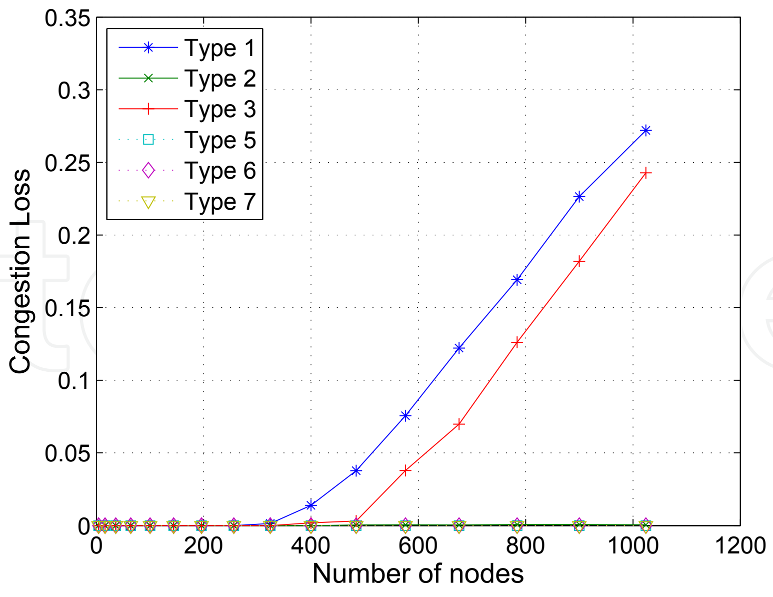

Fig. 10. Packets lost due to congestion for various traffic types. Shown in the figure is the fraction of packets lost due to congestion over all packets lost in transit.

reliability driven queries (Type 5), real time queries (Type 6), and mission critical interaction (Type 7). We analyze the average round trip times (RTT) for various kinds of queries into the network. Our workload generates queries to random motes in the network at various distances. For a 9-week long interaction, we summarize the interactivity times for networks at scale.

The interaction RTTs are plotted in Figure 11. Dynamic routing plays a major role in ensuring that interactivity times are kept low for real time queries (Type 6), acceptable for mission critical queries (Type 7) and relatively higher for reliability driven queries (Type 5). Coupled with high delivery ratios of Types 5 and 7, and short turn around for Type 6, we successfully meet the subtle variations in interactivity demanded by PdM.

5.7 Average Path Distribution

We finally characterize the path distribution statistics for various traffic types in the network (Figure 12). This simulation was run for a collection of 1024 nodes arranged using a 32x32 grid, with a 10 feet inter-node spacing. For every packet received at the base station, we measure the number of hops that it took build a frequency distribution for various hop counts. The curve is representative of route selection since each traffic type generates sufficient number of packets at various distances from the base station.

Requirements of PdM apart, nature of route selection is best captured in this plot. Reliable traffic (Types 1 and 5) take numerous short hops of high quality links, and register large hop counts. Real time traffic (Types 2 and 6), which is routed greedily based on shortest paths, takes the least number of hops. Mission critical data are offered hops that range in between reliable and real time traffic.

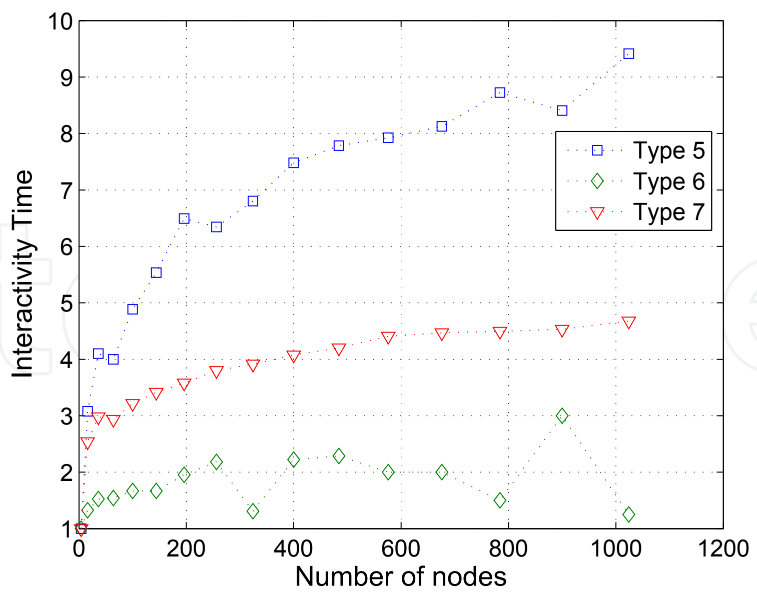

Fig. 11. Average round trip times for interactive queries with the deployment

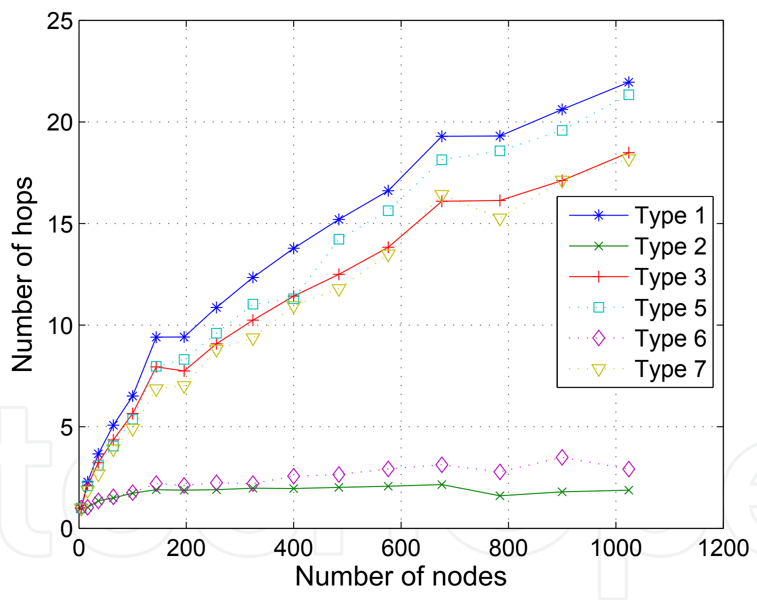

Fig. 12. Path distribution statistics for various traffic types for a deployment of 1000 nodes

6. Discussions

Exposing application requirements creates a plethora of in-networking possibilities. We show the impact of creating a dynamic network architecture with the use of the preamble bits at various levels of the stack: applications, protocol validation, energy efficiency, aggregation, fairness and differentiated services.

Application Programming: With data becoming self identifying, application programming is agnostic to the lower layers of the stack. Since the preambles are not protocol dependent, the scheme is guaranteed to work even when the mapping between the preamble and a particular protocol change over time. The framework in turn understands the nature and requirements of the payload, and accordingly wires a routing module to serve the purpose. We have diverged from priority based approaches, where our three bit scheme provides no notion of relative importance of a packet. We believe this is important, because the subjective notions of a packets relative priority are often debatable, inconsistent and prone to errors. Application programming is virtually error free, since it is not possible to confuse between a packets requirements, whereas it might be really hard to choose between a priority level of 5 or 6 for a range from 0-7 as in the case of DiffServ.
Protocol Validation: Protocols in sensornets are validated over a set of workload at least thought to be representative of the entire application domain. Most protocols are evaluated on a workload for which the protocol is optimized for. For example, a real time routing protocol is evaluated for a workload that emphasizes real time traffic alone. Most practical deployments would generate a workload of which real time communication is only a part of the requirement. Hence, a protocol's behavior in the face of real world deployment traffic is largely unknown. A dynamic routing framework, which can house various types of protocols optimized for various other types of traffic could form the basis of applying real-life workload to evaluate any alternative choice of protocol optimized for a given traffic type.
Energy Efficiency: Energy conservation has been an integral motive of almost every protocol proposed thus far. This trend in general has led to various “energy efficient" protocols with crippled communication abilities. Majority of energy drain happens at a nodes communication interface, and this trend shall continue to hold true well into the future. While computational subunits can be expected to improve in terms of energy per unit computation (e.g. Moore's Law), communication interfaces are governed by static laws of physics. Research by Pottie and Kaiser (21) shows that over 3000 instructions could be executed for the *same* energy cost of transmitting one bit wirelessly by 100 meters. The only foreseeable way to conserve energy is to *compute* more, and communicate *wisely*. With the application's requirements becoming visible, a whole host of in-network processing is now made possible to take the most appropriate action for every packet.
Aggregation: This domain has been widely studied in the sensornet domain, with excellent contributions in literature. However, aggregation cannot be abstracted as a component that generally applies to *any* payload. Aggregation comes with a little cost of delay in terms of processing, and in some cases, stalling for potentially related information to arrive. Delay sensitive data is generally not very amenable to aggregation.
Fairness: Presently, fairness in sensornets is not a well defined notion. Classical notions of fairness, where every player gets an equal share, needs a redefinition in the case of sensor nets. Not all nodes in the sensornet are the same, and neither are all packets equally important. The authors in IFRC (22) raise whether fairness is a reasonable initial design goal in a sensornet. While this may be difficult to answer without extensive deployment experience, what is generally lacking is a *basis* for defining fairness. For example, which packets should be transmitted in what order, or at what power level, or who should be dropped when congestion grows are questions that seek answers.
Differentiated Service: Traditional data networks passively transport bits from one end system to another. To the network, the payload is opaque as far as requirements are concerned, and the role of in-network processing is limited. Protocols and policies ought to act according

to the relative importance of a particular packet in question. Not all packets in a sensornet are of equal importance. For example, during times of congestion, dropping an arbitrary packet makes little sense: a packet carrying a critical alert information is clearly more important than a packet carrying regular sense-and-disseminate data. Similarly, a node with little energy might not receive mundane data, but might be willing to forward critical information when it offers a shorter path. Service differentiation is a strong incentive in sensor networks, largely because typical deployments are governed by higher level logic dictating requirements.
Richer Possibilities: The preamble bits and the dynamic framework provide a basis for adaptive protocols, allowing richer interactions with the deployment. It provides a powerful platform for user driven customization of the infrastructure, allowing new services to be deployed at a faster pace.

7. Conclusions

Typical deployments would consist of multiple concurrent applications, all of whose success leads to the fulfillment of a deployments objective. With every application placing its own subjective communication demand on the framework, there is an urgent need to both expose these requirements to the communication framework, and dynamically customize behavior for every type of application. We have presented a simple scheme of using just three intent bits to completely describe communication patterns the stack, and we use this to drive a dynamic routing framework that customizes its routing behavior for every packet type in the system. We have proved its effectiveness in meeting the demands of a fairly complete deployment of industrial monitoring using PdM, where we analyzed behavior at scale for thousands of nodes, and implemented a prototype of a 40 node wireless testbed.
Diversity in application requirements for sensornets has led to an explosion of network protocols. Protocol developers focus performance for a particular traffic type, and likewise validate protocols for that type of traffic. Our framework allows for rapid protocol development, integration and validation in the face of realistic workloads. With a need to emphasize performance, developers further make assumptions about interfaces and functionalities that further limits synergy across research efforts. In our quest to build a configurable framework, we have regularized interface assumptions to distill core protocol features as individual components. This would ensure that the core components can evolve independently, and research efforts on any component can be seamlessly ported across deployments.
The role of in-network processing is currently limited in sensornets. With the application requirements made visible to the stack, there is great potential to design application specific processing at every node. Our dynamic routing is just one example of using the requirements to switch routing behavior at the network layer. In general, there is excellent potential for designing medium access protocols, scheduling protocols, congestion control algorithms and energy efficiency modules at various layers of the stack using the preamble bits.

8. References

[1] D. Braginsky and D. Estrin. "Rumor routing algorithm for sensor networks", *Proc. First ACM International Workshop on Wireless Sensor Networks and Applications, (WSNA)*, Sept 2002.
[2] Q. Cao, T. Abdelzaher, T. He, and R. Kravets. "Cluster-Based Forwarding for Reliable End-to-End Delivery in Wireless Sensor Networks", *IEEE Infocom*, May 2007.

[3] T. E. Cheng, R. Fonseca, S. Kim, D. Moon, A. Tavakoli, D. Culler, S. Shenker, and I. Stoica. "A modular network layer for sensorsets", *Proc. 7th Symp. on Operating Systems Design and Implementation (OSDI)*, Nov. 2006.

[4] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher. " Real-Time power-aware routing in sensor networks", *Proc. IEEE International Workshop on Quality of Service (IWQoS)*, June 2006.

[5] D. Culler, P. Dutta, C. T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao. "Towards a sensor network architecture: Lowering the waistline", *HotOS X*, June 2005.

[6] D. D. Cuotu, D. Aguayo, B. Chambers, and R. Morris. "Performance of Multihop Wireless Networks: Shortest Path is Not Enough", *First workshop on Hot topics in Networks (HotNets-I)*, Oct. 2002.

[7] D. S. Couto, D. Aguayo, J. Bicket, and R. Morris. "A High-Throughput Path Metric for Multi-Hop Wireless Routing", *ACM Mobicom*, Sept 2003.

[8] A. Dunkels, F. Osterlind, and Z. He. "An adaptive communication architecture for wireless sensor networks", *ACM Sensys*, Nov. 2007.

[9] R. Fonseca, S. Ratnasamy, J. Zhao, T. E. Cheng , D. Culler, S. Shenker, and I. Stoica. "Beacon-Vector Routing: Scalable Point-to-Point Routing in Wireless Sensor Networks", *Proc. Usenix NSDI*, July 2005.

[10] J. L. Gao, "Energy efficient routing for wireless sensor networks", Ph.D. thesis, Electrical and Computer Engineering Department, UCLA, June 2000.

[11] T. He, J.A. Stankovic, C. Lu, and T. Abdelzaher. "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks", *Proc. ICDCS'03*, May 2003.

[12] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. "Energy-efficient communication protocol for wireless microsensor networks", *Proc. of 33 Hawaii International Conference on Systems Science (HICSS)*, Hawaii, Jan 2000.

[13] N. C. Hutchison and L. L. Peterson"The X-Kernel: An Architecture for Implementing Network Protocols", *IEEE Trans. on Soft. Engg.*, 17(1), Jan. 1991.

[14] C. Intanagonwiwat, R. Govindan, and D. Estrin. "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", *ACM/IEEE Mobicom'00*, Aug 2000.

[15] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman and M. Yarvis. "Design and deployment of industrial sensor networks: experiences from a semiconductor plant and the north sea", *ACM Sensys*, Nov. 2005.

[16] P. Levis and D. Culler, "Mate: A Tiny Virtual Machine for Sensor Networks", *Proc. Intl. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Oct 2002.

[17] K. Nichols, V. Jacobson, and L. Zhang. "A Two-bit Differentiated Services Architecture for the Internet". Internet Engineering Task Force, RFC 2638, July 1999.

[18] S. W. O'Malley and L. L. Peterson. "A dynamic network architecture", *ACM Transactions on Computer Systems (TOCS)*, 10(2), May 1992.

[19] S. Pattem, B. Krishnamachari, and R. Govindan. "The Impact of Spatial correlation on Routing with Compression in Wireless Sensor Networks", *ACM/IEEE IPSN*, April 2004.

[20] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica, "A unifying link abstraction for wireless sensor networks", *ACM Sensys*, Nov 2005.

[21] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors", *Communications of the ACM*, Vol. 43(5), May 2000.

[22] S. Rangwala, R. Gummadi, R. Govindan, and K. Psounis. "Interference-Aware Fair Rate Control in Wireless Sensor Networks", *ACM Sigcomm*, Sept 2006.

[23] D. Sharma, V. Zadorozhny, and P. Chrysanthis. "Timely data delivery in sensor networks using whirlpool", *Proc. 2nd international workshop on Data management for Sensor Networks*, Aug. 2005.

[24] F. Stann and J. Heidemann, " RMST: reliable data transport in sensor networks", *First IEEE Intl. Workshop on Sensor Network Protocols and Applications (SNPA)*, May 2003.

[25] M. Venkataraman, K. Muralidharan, and P. Gupta. "Designing New Architectures and Protocols for Wireless Sensor Networks: A Perspective", *IEEE Secon*, Sept 2005.

[26] C.Y. Wan, S.B. Eisenman, and A.T. Campbell. "CODA: Congestion Detection and Avoidance in Sensor Networks", *ACM Sensys*, 2003.

[27] A. Woo, T. Tong, and D. Culler. "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks", *ACM Sensys*, 2003.

[28] C. Y. Wan, A. T. Campbell, and L. Krishnamurthy. "Pump-slowly, fetch-quickly (PSFQ): a reliable transport protocol for sensor networks", *IEEE Journal on Selected Areas in Communication (JSAC)*, 23(4), pp. 862–872, April 2005.

[29] M. A. Youssef, M. F. Younis, and K. Arisha. "A constrained shortest-path energy-aware routing algorithm for wireless sensor networks", *Proc. of IEEE WCNC*, March 2002,

[30] Y. Yu, L. Rittle, J. LeBrun, and V. Bhandari. "MELETE: Supporting Concurrent Applications in Wireless Sensor Networks ", *ACM Sensys*, Nov 2006.

[31] J. Zhao and R. Govindan. "Understanding Packet Delivery Performance In Dense Wireless Sensor Networks", *ACM Sensys*, Nov 2003.

[32] University of California, Berkeley. TinyOS CVS Repository at SourceForge. `http://sf.net/projects/tinyos`. June 2007.

[33] MicaZ motes specification. `www.xbow.com/products/ product_pdf_files/ wireless_pdf/6020-0060-01_a_micaz.pdf`