# Simulated Analysis of Connectivity Issues for Sleeping Sensor Nodes in the Internet of Things

Tyler Ward, Kirk Martinez, and Tim Chown
Web and Internet Science, Electronics and Computer Science
University of Southampton
Southampton, United Kingdom
t.ward@ecs.soton.ac.uk, km@ecs.soton.ac.uk, tjc@ecs.soton.ac.uk

## ABSTRACT

The growth in wireless sensor network deployments requires a move towards more standardised systems to improve compatibility and to reduce development times. The technologies being developed as part of the Internet of Things, such as 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks), can greatly assist with this aim. Connecting low power wireless sensor network devices to the Internet of Things presents certain challenges. One of these challenges is the lack of constant connectivity to sensor nodes with sleep states. Current internet technologies expect that devices are always contactable which is not the case in sensor networks. We simulate and evaluate several solutions to this problem in a multitude of different scenarios. We conclude that delay tolerant networking is an effective solution to the challenges created when dealing with sleep states while minimising overheads. However, current standardised delay tolerant technologies are not easily applicable for use with sensor networks, so a new standard needs to be created to meet the requirements described in the paper.

## Categories and Subject Descriptors

C.2.0 [**Computer-communication networks**]: General—*Data Communications*

## Keywords

Sensor Networks; Internet of Things; Wireless Sensor Networks; Delay Tolerant Networking

## 1. INTRODUCTION

In the past, wireless sensor networks have often used customised hardware, software and network protocols for each deployment[1]. The continuing growth in the need for wireless sensor network deployments limits the sustainability of this approach. This methodology also causes problems when trying to connect sensor networks to each other, or other devices to sensor networks [5]. One example is future home networks with devices connected over a mixture of low power and traditional links. Connecting sensor networks to the Internet and making them part of the Internet of Things is one solution to this problem. This also has the potential to unlock additional capabilities using the improved connectivity. Much work has already been done under the Internet of Things banner to make this a possibility[8].

However, in order to benefit from migration to the Internet of Things there are several areas which still need further research in order to support the requirements of wireless sensor network devices. Many sensor nodes need to spend most of their time in low power sleep states with communications turned off[1]. Although improvements in low power radio technology are allowing greater use of the radio for the same sensor lifetime[4], these improvements are unlikely to alleviate this issue entirely, especially for nodes that will remain battery powered for years. This restriction is incompatible with the expectation of continuous end-to-end connectivity as assumed by the majority of Internet connected systems.

In previous non IP (Internet Protocol) systems this problem is handled by an application level gateway which could respond in various ways, such as informing the enquirer that the node is asleep or serving cached data. With current IP systems when a host attempts to communicate with a device that is in a sleep state the message is lost as it cannot be delivered to its destination. There are methods in place to allow senders to be informed that a message could not be delivered, such as the destination unreachable message, however, this only gives a rough idea of why a node is unreachable. There is no guarantee that such a message would even be sent or that they will not be dropped by the network.

Whilst the issue of communicating with sleeping nodes has been identified as a challenge[2, 13], as yet there has only been a limited amount or research into this area. Current solutions rely on the node initialising all communication with the outside world, with the inherent expectation that the rest of the world is always accessible[13].

## 2. IMPROVING COMMUNICATIONS WITH SLEEPING INTERNET OF THINGS DEVICES

Traditional wireless sensor networks are not usually expected to receive traffic from Internet devices. However, the Internet of Things has the expectation of the same bidirectional communication capabilities that also exist on the non constrained Internet.


*PE-WASUN'14,* September 21–26, 2014, Montreal, QC, Canada.

ACM 978-1-4503-3025-1/14/09 ...$15.00.
http://dx.doi.org/10.1145/2653481.2653490.

There are many different solutions to overcome the communications issues faced when dealing with sleeping Internet of Things devices. All of the current potential solutions come with an attached overhead which needs evaluating. Many of the solutions require firmware changes on the devices or software changes in the software interfacing with the devices. Sleeping nodes are not yet commonly deployed on the Internet so there is little backwards compatibility that needs to be maintained. There are existing software tools for management and control which can not be modified easily for which maintaining compatibility would be advantageous.

## 2.1 Node Initiates all Communication

The current suggested solution to this issue is to rely on the node to start all communication[13]. This avoids the issue of needing to know when the node is on. Whilst this method is appropriate in certain circumstances, it imposes limitations that can hinder certain tasks.

Where a node needs to have changes to its configuration parameters performed remotely it will need to periodically poll a control server to check if updates are required. It is worth noting that as the node initiates the communication, what it needs to talk to and how often becomes part of this configuration. The frequency of this checking becomes an important decision, too frequently and the node uses unnecessary power, too infrequently and changes to the node's configuration take a long time to apply.

In the situation where communications are not reliable or are not guaranteed to be available, there is the potential for wasted power. For example, A relay node may be unable to relay traffic due to power loss, or damage. During this period the node will need to continue to attempt to communicate as it will be unable to tell if there is a working connection until it sends a message. In environmental sensor networks such as Glacsweb[9] these communication outages can last many months. By having the request come to the node it will avoid transmitting when there is no connection as no requests will be received. A similar scenario can occur if systems are no longer interested in data from a node for a period of time. If the system either does not or is unable to inform the node this is the case, then the node will not be aware and will continue to transmit data that is not being utilised.

One scenario where this has a significant ease of use benefit over other technologies is when the node is behind access controls such as firewalls or NATs (Network Address Translations). For this reason this method is used in many Internet of Things smart home devices. They phone home using IPv4 to avoid users needing to make network configuration changes. The use of IPv6 can remove the limitations of NAT, facilitating easier communication between devices, however, IPv6 has not yet reached the stage of widespread adoption [3]. Firewalls will still need configuring for use with IPv6 however the user will have more control than with IPv4 firewalls using NAT. It should be noted that in several cases such as Violet and their Nabaztag product the phone home server for such devices has been shut down rendering them inoperable [17].

## 2.2 Polling the node

The most basic solution available is to repeatedly send requests until a response is received. This is a particularly bad solution with regards to conserving sensor network resources. In order to catch the node when it is awake, the host needs to poll the node frequently enough so that the node cannot wake up and go to sleep in between messages. The nodes will need a reasonable amount of on time to avoid needing to poll too frequently. Even with a long amount of time with the radio on, this method generates a lot of unnecessary traffic. This traffic will impose an increased load on other nodes that are awake as they will need to evaluate whether that packet is for them or if they should forward it on. Many systems doing this will consume a significant proportion of a low bandwidth communication link, this is especially true when the bandwidth bottleneck is the sensor itself rather than the channel bandwidth as the sensor may miss other important messages.

This method does not require any changes to be made to either the network hardware or to any node firmware. Providing that the software on the computer making the queries can be configured to tolerate intermittent connections then there should not be an issue. Many pieces of software, however, are unable to be configured like this.

## 2.3 Time Coordination Between Data Sources and Sinks

If the periods when a node is able to receive communications are known in advance then it is possible for hosts that want to request data from a sensor node to time their requests so that they fall within the communications window. This solution does not require any additional equipment or always on devices.

The major downside to this solution it that the software running on the remote host has to be configured with the node's schedule. Software that is unable to be scheduled or nodes with unpredictable schedules cannot utilise this approach.

In order for the devices to keep their schedules in sync both devices need to maintain an accurate clock. Depending on the capabilities of the sensor device it may also be necessary to add an RTC (Real Time Clock) module to maintain an accurate enough time. A sensible method of maintaining the clock is to use NTP (network time protocol) or similar clock synchronisation protocols. Depending on the quality of the RTC this could be done quite infrequently, this adds some additional network traffic and processing but not a huge amount. This method also requires that a node is powered up for long enough to accommodate any time drift between them and the remote host.

## 2.4 Transparent Proxy

With some sensor systems a proxy server can be used in order to make it appear as if the nodes are always on. With this solution, when the hosts request the data from the node they are instead sent to a proxy server. This server can forward the request for the data when the nodes are on-line and store the responses for later. When the nodes are off-line it can reply with cached copies of the data. Some protocols such as CoAP (Constrained Application Protocol) have inbuilt support for this function [16]. Although the proxy can only understand some types of request, unrecognised requests or communication protocols can be passed onto the network to provide compatibility with other systems which are on the same network.

Multiple requests to a single node for data can be responded to without the need to request data from the node

again. There is a risk of data not being refreshed or not being available if data is not requested while a node is on. This can be solved by the proxy automatically updating its cache of data by talking to the nodes itself, however, with this the proxy begins to act more like the data distribution system solution.

Messages to control a node will need to be passed through the proxy in order to be received by the node. This means that those messages will need to be correctly timed in order to be acted on.

## 2.5 External Data Distribution System

On the majority of non Internet connected wireless sensor networks, the data from the sensor nodes is gathered together in one place where it can be requested by networked devices. While this was almost essential in previous systems the Internet of Things mainly relies on direct node communication which will scale badly when there are more interested parties. By having a device perform this data management it can drastically reduce the complexity required for the majority of devices. As they are requesting the data from an always on system they no longer need to worry about communication with sleeping nodes. Although this solves the problem for the majority of the hosts it does not solve the problem for itself. As there is usually a single data store, it could have the nodes push data, or could request data from the nodes using another of the solutions discussed.

With this solution other hosts are limited to the data collected by the store and the rate at which the data store collects it. As the data store needs to understand how to request the data from a node there is a requirement that the data store and node share a common protocol and data format. Multiple hosts interested in data from a single sensor do not increase the amount of communications to a node.

The server can also convert the data into multiple formats and consolidate sensors with otherwise incompatible data formats or protocols into a common format presented to interested parties. In order to do this the data distribution server needs to understand the format of the data in order to present it to other interested parties.

There may initially seem to be little difference between using a data distribution system and the existing proprietary systems. While a data distribution system might serve the majority of requests for data on a network, other IP based systems which would like direct access can still achieve this. A caveat with data distribution systems is that while they are suitable for allowing many devices to get data from a system they are unable to handle sending messages the other way, thus making this solution redundant for actuator nodes.

## 2.6 Message Queues

Message queues are commonly used in systems where multiple processes need to pass information to one another either on the same machine or somewhere else on the network. This technique could also be used to coordinate communication between sensor nodes[7].

In this set-up, nodes would operate by pushing data to a queue on a central machine whilst also checking for any messages that may be waiting for it. Hosts can subscribe to the queues published by the node, in order to send a message it can be added to the nodes queue to be collected later. The data sent would encapsulate the higher level protocol or raw data in a message queue packet, this breaks the end to end IP connectivity that is gained by moving to the Internet of Things.

Similar to data distribution systems and proxies the message queues allow for many data consumers without increasing load on a node. Unlike data distribution systems however it can also handle messages being sent to a node. In order to do this a node will need to send a message to find out if there is a message waiting for it, although this can also be done as part of sending data in.

## 2.7 Delay tolerant networking

A common solution in non Internet connected sensor networks is to use store and forward technology to allow data to be sent when a link is reestablished. IP does not have support for these features, although it can be added at a higher layer usually referred to as delay tolerant networking.

While it is possible to simply hold the messages intended for a node until it wakes up, issues begin to arise as the times between on periods increase. As the time extends beyond a few seconds hosts will assume that the message has been lost, in order to solve this, there needs to be some information sent with the message in order to define how it should be handled. For example, most messages sent to sensor networks have a finite time in which they are relevant, after said time there is no point in sending them any more as all they will do is use unnecessary bandwidth and battery life.

The delay tolerant Bundle Protocol[15] can encapsulate IP traffic to allow store and forward operation. This encapsulation adds another layer that needs to be understood and decoded in order to process the message. Although the Bundle Protocol has successfully been ported to sensor networks[14] it is not a perfect solution. It is designed to support a large range of protocols and results in a long header; this is obviously not ideal for use in power constrained devices.

A potential improvement to this solution would be to have a delay tolerant IP header that can allow equipment supporting delay tolerant features to handle the packet appropriately without the need for significant extra overhead. A useful feature of using an IP header over adding an additional layer is that if a node cannot understand the header it can still process the message thus removing many compatibility problems that would otherwise arise.

# 3. TESTING SCENARIOS

Due to the scope of potential Internet of Things sensor deployments the potential solutions will need evaluating in a mixture of different practical network scenarios.

We review five scenarios, with figures 1 to 5 showing example layouts of what the different network topologies look like.

## 3.1 Real-time Sensing Network

In many cases the data from sensors is needed for processing as soon it is recorded. Examples of such sensors are thermostats and water level sensors[6] where their data is used to keep an environment or system within certain parameters.

Some sensors of this type will be always-on devices, however, this will cause battery powered nodes to consume their energy reserves faster. In order to extend the battery life, these sensors can turn on for short periods regularly to

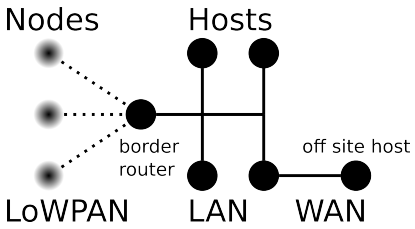


**Figure 1: example real time network**

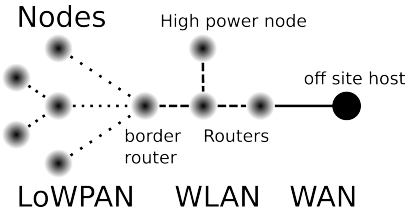


**Figure 2: example environmental network**

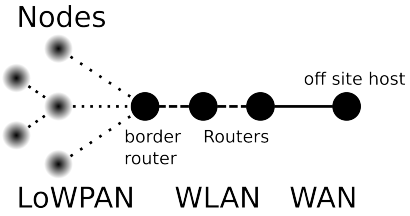


**Figure 3: example event detection network**

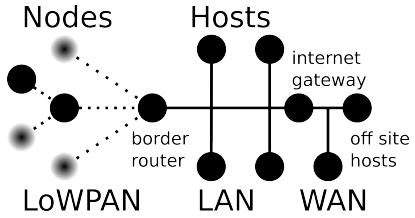


**Figure 4: example mixed network**

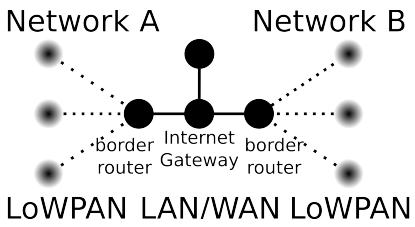


**Figure 5: example multiple network setup**


**Key for the network layouts**

record and transmit data and spend the rest of the time in a sleep state. These networks are likely to have a reliable connection to the wider Internet with a border router that is probably an always on device.

## 3.2 Environmental Sensing Network

Like the real-time sensing networks these networks will be recording data about their environment, however, there is no need for the data to be processed immediately [10]. This allows the data to be reported back in chunks with longer intervals in between. This saves power as the radio systems are used less frequently.

These 'log and send later' sensor nodes are regularly used when studying the environment or recording data for historical or evaluation purposes. Often replacing the battery on these types of device is difficult or even impossible.

Networks of this type often have routers which are themselves power constrained and likely to need sleep states in order to conserve power. There may be several of these low power routers between the sensor network and the Internet for networks in remote locations.

## 3.3 Event Monitoring Network

These sensors are intended to report events back as they happen rather than polling the environment. Examples of these sorts of networks are detection of natural hazards such as landslides or volcanic activity[18], similar principles would also apply to devices in more everyday scenarios such as fire alarms or security systems.

While these networks will usually push information when events happen there are many cases where it is necessary to communicate with the node, for example, to configure its sensitivity or confirm that it is still working. Such networks are likely to have an always on base station and attached communication infrastructure in order to ensure messages are relayed to their destination and never lost.

## 3.4 Mixed Networks

Due to the flexibility of the Internet of Things, a single network may include devices with different or multiple capabilities. An example of this is a HVAC (Heating Ventilation and Air Conditioning) system that shares a network with a security system.

Multiple network types sharing common infrastructure reduces the cost of providing multiple types of sensing systems or changing an existing deployment as new routers aren't required. In these networks sleeping nodes may coexist with non sleeping nodes that are not so power constrained. In such networks, the border router and its upstream connectivity will need to cope with the requirements of all of the nodes on the network, this will mean that it is probably going to be an always on device.

## 3.5 Internetwork Communication

The Internet of Things allows for far greater flexibility compared to existing sensor networks for machine to machine communications [19]. Internet technologies make it easier for devices to communicate to other devices inside the same network or in another network. There is a major issue when trying to do this for sleeping devices as the schedules for two devices might be completely different.

An example of this type of network could be an irrigation control system. An intelligent device that informs a user if a

**Table 1: Solution applicability in different environments**

| | Test cases | | | | |
|---|---|---|---|---|---|
| Solution | Real-time Sensing | Environmental Monitoring | Event Monitoring | Mixed Network | Multiple Networks |
| Polling | Yes | Yes | Yes | Yes | No |
| Time Synchronisation | Yes | Yes | Yes | Yes | No |
| Transparent Proxy | Yes | No | No | Partial | No |
| Data Distribution System | Yes | Yes | No | Partial | Yes |
| Message Queues | Yes | Yes | Yes | Yes | Yes |
| Delay Tolerant Networking | Yes | Yes | Yes | Yes | Yes |

sluice gate is in the wrong position might use data from the network on the field as well as data from a sensor network monitoring water flow on a nearby river.

With multiple sleeping nodes direct communication becomes impossible if devices are running on different schedules. This communication could be coordinated by devices on the wider Internet or a local server. These solutions would require a reliable Internet connection or stable power supply respectively in order to operate, which might not be practical in extremely remote areas. Alternatively, using delay tolerant networking border routers could buffer packets until the other device is contactable.

## 3.6 Solution Applicability

While some of the solutions proposed earlier will work with any of the test cases, many are situational and either provide no benefit or cause issues for other test cases. For example, a proxy setup provides no benefit for environmental sensor networks as each data upload is different and ideally is only transmitted once.

Table 1 shows the solutions that will work with different test cases. Although some of these are obviously bad solutions, such as polling for a node that only turns on for a few seconds each hour or day, these were still included for comparison.

# 4. SIMULATION SOFTWARE

To evaluate all of the potential solutions in the appropriate test cases using real sensor network hardware would take a considerable amount of time and resources. Although the nodes could be emulated at a low level using tools such as Cooja [12] to avoid hardware costs this would still require a significant investment of time for the creation of device firmware. Instead a network level simulator was used, which simulates the sensors at a high level avoiding the complexity of hardware interfaces and encoding or decoding software protocols. Using a simulator that only focusses on the interested layers to simulate the nodes will still produce realistic results.

Different nodes may have different application layers and sensing capabilities which will put different requirements on a node. It has been assumed that the effect on the node from the higher layer protocols and sensing cycles can be broken down into a fixed load that the node will perform regardless of what happens on the network, and an additional load for every request. As the fixed load is independent of the effects of the network it can be ignored. The additional load introduced for each request is assumed to use the same amount of resources and as such can be included as part of the load for the node to process a message.

Any alterations to the network layer are not going to affect the way the lower layers or the underlying hardware will behave. As a result it can be assumed that the load imposed by the lower layers for any volume of network traffic, will be the same as any other traffic.

The changes to some of the node systems as required by solutions such as message queues and delay tolerant networking will add some additional load onto the node. Given how little of the total energy required is needed to service the packet handling logic for a request, any changes to this logic required to manage different solutions will probably have a negligible power impact. The additional bytes that might need to be transferred for those systems, however, will increase power usage of the radio which will need to be taken into account.

There will also be some load introduced onto the node whenever a packet is detected by the radio, the packet might not be intended for that node but the node will still need to confirm this. With these assumptions the total load of the node can be evaluated by using the amount of packets a node sends and receives, the other loads on the node are proportional to this or are a fixed drain on resources.

As all of the elements in the system can be modelled as events the simulator was created using the SimPy[11] discrete event framework. Each of the elements in the simulation would be triggered by events from either an internal timer or from other events such as local processes or network activity.

In addition to simulating the end devices on the network the links and routers between devices will also be simulated. The simulations will model the flow of every packet across the network and record important metrics about the packet's transmission. Factors such as latency, jitter and packet loss are taken into account at each hop across the network. The transmission time is modelled using an average transmission time with a jitter added using a standard deviation model. Packet loss is modelled by randomly dropping packets based on the link's loss rate. For wireless links there can only be one message on the channel at one time. This is modelled by delaying transmissions until the channel is clear, representing the use of a clear channel assessment system.

The solutions were separated into two categories, those that altered the timing of messages and those that added features. Along with the timings that would be a potential solution, additional timings were added, for example, ones chosen randomly or offset by a small amount of time. These timing options can then be used when testing the network with and without different additional features.

The simulator records information about if and how quickly each of the devices are responding to requests as well as the

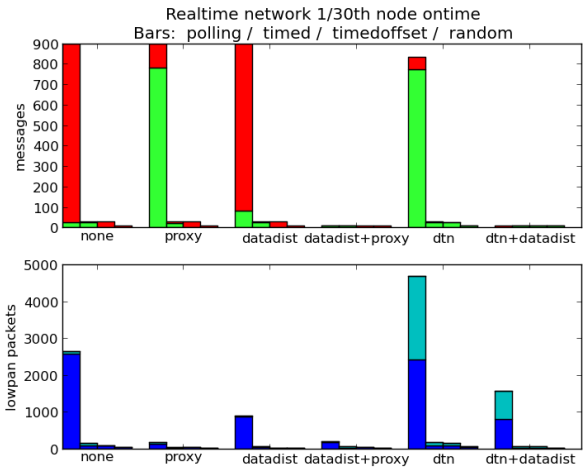


**Figure 6: Success rates and quantity of messages sent on a real-time sensing network.**

amount of traffic being sent over each of the network connections. The simulations can be run on each of the test cases to determine the effectiveness of solutions with different network and sensing configurations.

## 5. SIMULATION RESULTS

The two main statistics that were used for initial analysis were the amount of network activity generated on the lowpan network and the success or failure of queries sent by other devices.

The amount of network traffic used to communicate with the nodes is a good measure of how effective the solutions are, providing that these messages are also successful. If the required communication can be achieved with less traffic on the lowpan network then less power will be needed to send and receive thus allowing a greater sensor life and reducing network congestion.

Figure 6 shows multiple potential solutions on a real-time sensing network with a node on for a 30th of the time for 15 node cycles, for example a node on for 10 seconds every 5 minutes for an hour and a quarter. The data distribution server was set up to collect data using the request timings that would normally be used by the hosts. The hosts collecting data from the server were instead set to use a random request schedule as hosts using a data distribution system shouldn't need to use specific timings.

The Y axis is the amount of requests for data and message passed on the lowpan network. The request count bar shows how many requests where sent from hosts and how many where responded to. Light green indicates a response was received, the darker red that no response was received. The lowpan packet count bars show the directionality of the data, the lighter blue was sent by a node where as the darker blue was sent from the border router. The bars are grouped based on what features are enabled on the network, labelled on the X axis. Within those groups they represent the request timing used, labelled above the graph.

As would be expected, polling causes a massive amount of network traffic on a real-time sensing system when compared to the other solutions. This will increase as the percentage of time when the node is off increases such as in environmental sensor networks. Timed coordination of nodes works well when the schedules are aligned, however, should they become out of synchronisation then communications to the node are completely lost until the devices can regain synchronisation.

Figure 7 shows an example amount of traffic on the test case networks during a day. The polling solution has been removed from this graph to allow easier visibility of the other solutions. The colour scheme and layout is the same as used in Fig 6. The small amount of failed messages encountered with the majority of systems was due to the packet loss programmed into the simulation, this packet loss was modeled at 5% for the lowpan network with regular wireless links at 1% and wired links at 0.1%.

In the case where many hosts are wanting to request data from a single node the systems that could redistribute data had a major impact on the amount of data. In order for the proxy server to operate it requires requests that are correctly timed in order to update its cache, if nothing is requesting on the correct schedule then it will be unable to service requests that fall outside those times. Data distribution systems while effectively managing the amount of queries being received from other devices, were only truly effective when combined with other solutions. Message queues also resolved this issue, however, they have the unfortunate requirement that messages need to be in the message queue protocol, this breaks the end to end IP level connectivity and hence removes a lot of the benefits from moving to an Internet of Things platform. Nodes also need to query the message queue to determine if there are any messages for them. While this can be included when sending data to the message queue, should no data be being sent then a separate message will be required.

In the multi network test case it can be seen that only those solutions able to buffer either data or packets are able to maintain communications with multiple data providing nodes due to the mismatched schedules. Alternatively it would be possible to have the node fetching the data turn on for each schedule to request the data, this requires that the node knows all connected schedules and requires the node to use more energy. Schedules could be aligned so that there is a single schedule for all data sources, this avoids requiring additional wake cycles for the node requesting the data. Unfortunately this solution imposes quite a few constraints on those deploying a network and as such should not be relied upon.

Delay tolerant networking universally improves the situation for sleeping nodes allowing reliable communications without the need for end to end time synchronisation. In order to describe how the packets should be treated, and how much delay is acceptable, extra information needs to be transmitted with the packet. This could be done using IP headers, this will add a few additional bytes to any message sent with this feature, messages that do not require this will not have any additional overhead. While it is possible to implement delay tolerant features without adding any overhead this gives no information about how packets may be delayed and can lead to situations like the one that follows.

It was found when hosts were trying a polling style request with delay tolerant networking enabled and the node off period was extended, that the success rate of requests would drop off. This is caused by the traffic from the entire

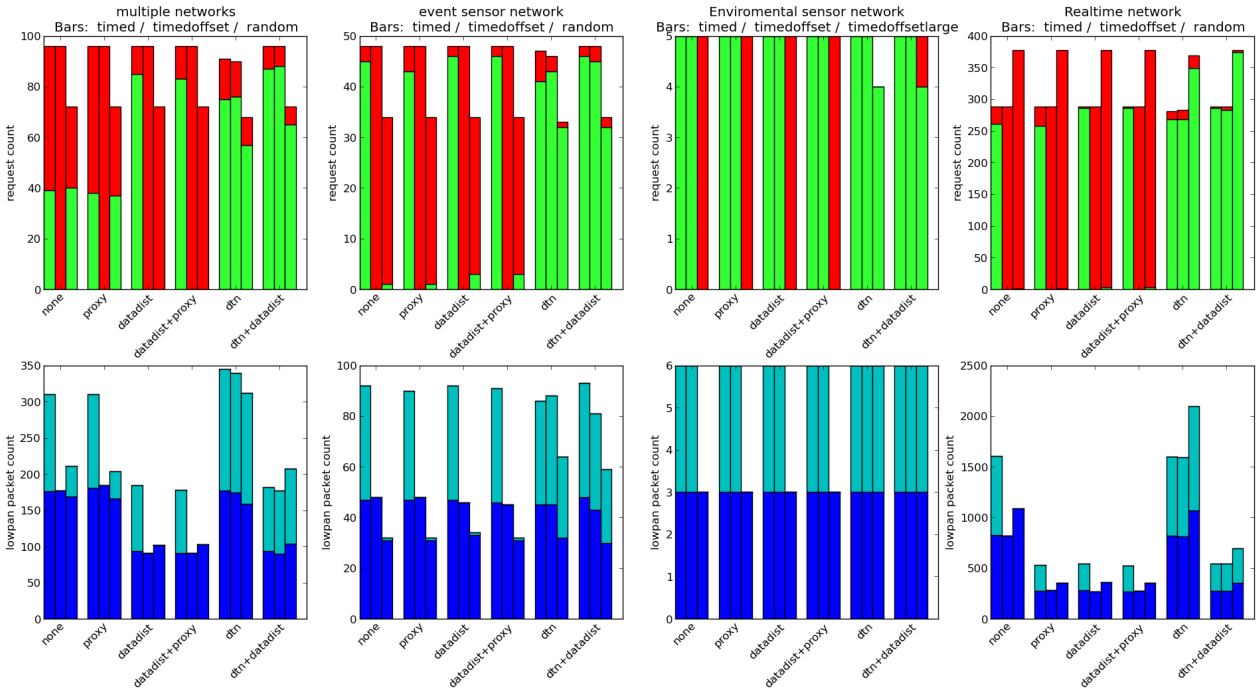


**Figure 7: Network utilisation during a 24 hour simulation modelled with different test cases and solutions**

period when a node is off being condensed into the nodes on period, causing the network to become overloaded similar to a denial of service attack. This emphasises the need for appropriate precautions to be taken such as detailing the useful life of a packet to prevent such an issue whether accidental or deliberate. With the bandwidths available to a single normally networked host it would be easy to disrupt a large number of lowpan networks.

In order to forward the packets at the correct point the border router with delay tolerant capabilities will need to know when the node is on-line. Depending on the network technology and configuration used this can be done in one of several ways. Most of these methods will not require any additional communication from the node. The router can be informed of the node's schedule as in the time synchronisation solution and use this to know when to forward the messages. This obviously inherits the synchronisation issues faced by the timed solution, however, due to their proximity maintaining synchronisation of clocks and schedules is an easier task and could potentially be included in a DTN header.

For some networks the node will need to communicate with the border router in order to connect to the network when it turns on. The border router can use this to detect that the node is able to receive communications. As the router is on the same network as the sensor node it can detect any communications the node may initiate, even if those messages do not leave the network, as a way of determining if the node is on-line. If none of these are available it is also possible for the node to inform the border router when it turns on similar to message queues. However, this will be more efficient than message queues as the communication delay to the border router will be reduced, in addition should the border router not have upstream connectivity the message can still be collected providing it has already made it to the router.

# 6. CONCLUSIONS AND FUTURE WORK

As the need for very low power sensor devices connected to the Internet of Things increases, the ability to communicate with devices in sleep states will become increasingly important. While it is possible for nodes to initiate all communication this limitation will restrict what can be achieved with the network. Needing to poll for configuration changes leaves them vulnerable to having an invalid or out of date configuration, which may waste power.

Delay tolerant networking has been shown to have the capability to massively reduce issues with communications between nodes and outside services due to the effects of sleeping devices in sensor networks. Although existing delay tolerant protocols do not lend themselves for use on constrained IP based sensor networks due to their size and the fact that they encapsulate the message, the principles can be used to develop protocols which are. In order for these protocols to be efficient they will need to use a minimum of bytes as well as avoiding any extra communications wherever possible.

The next stage will be to develop and evaluate using IP based delay tolerant networking technologies on real world networks and systems. This will be compared with existing technologies and deployments through field trials and further simulation at a lower level. During this a design for a lightweight delay tolerant networking header suitable for use on both constrained and unconstrained networks will be developed. This protocol will need to be designed to minimise the amount of bytes transmitted, from both bytes added to

messages and new messages, in order to operate efficiently on constrained networks, whilst still having the capabilities available in the larger implementations. It should also allow end devices and intermediate routers which do not understand the protocol to be able to process the message. It is believed that the use of IP headers to add the delay tolerant features will achieve these requirements.

## 7. ACKNOWLEDGMENTS

The authors would like to thank the teams from the Glacsweb and Mountain sensing projects.

## 8. REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications magazine, IEEE*, 40(8):102–114, 2002.

[2] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010.

[3] J. Czyz, M. Allman, J. Zhang, S. Iekel-Johnson, E. Osterweil, and M. Bailey. Measuring ipv6 adoption. Technical report, Technical Report TR-13-004, ICSI, 2013.

[4] A. Dunkels. The contikimac radio duty cycling protocol. 2011.

[5] M. Durvey, J. Abeillé, P. Wetterwald, B. O'Flynn, Colin Leverett, G. Eric, M. Vidales, M. Geoff, N. Tsiftes, N. Finne, et al. Making sensor networks ipv6 ready. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 421–422. ACM, 2008.

[6] J. Gustafsson, H. Mäkitaavola, J. Delsing, and J. Van Deventer. Integration of an ip based low-power sensor network in district heating substations. In *The 12th international symposium on district heating and cooling*, 2010.

[7] U. Hunkeler, H. L. Truong, and A. Stanford-Clark. Mqtt-s a publish/subscribe protocol for wireless sensor networks. In *Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008. 3rd International Conference on*, pages 791–798, Jan 2008.

[8] L. Mainetti, L. Patrono, and A. Vilei. Evolution of wireless sensor networks towards the internet of things: A survey. In *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*, pages 1–6, Sept 2011.

[9] K. Martinez, P. J. Basford, D. D. Jager, and J. K.Hart. Poster abstract: Using a hetrogeneous sensor network to monitor glacial movement. In *10th European Conference on Wireless Sensor Networks*, February 2013.

[10] K. Martinez, J. K. Hart, and R. Ong. Environmental sensor networks. *Computer*, 37(8):50–56, 2004.

[11] K. Muller and T. Vignaux. Simpy: Simulating systems in python. *ONLamp. com Python Devcenter*, 2003.

[12] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 641–648, nov 2006.

[13] B. Ostermaier, M. Kovatsch, and S. Santini. Connecting things to the web using programmable low-power wifi modules. In *Proceedings of the Second International Workshop on Web of Things*, page 2, 2011.

[14] W.-B. Pottner, F. Busching, G. von Zengen, and L. Wolf. Data elevators: Applying the bundle protocol in delay tolerant wireless sensor networks. In *Mobile Adhoc and Sensor Systems (MASS), 2012 IEEE 9th International Conference on*, pages 218–226, 2012.

[15] K. Scott and S. Burleigh. Bundle Protocol Specification. RFC 5050 (Experimental), Nov. 2007.

[16] Z. Shelby, K. Hartke, and C. Bormann. Constrained Application Protocol (CoAP). Active Internet-Draft, June 2013.

[17] J. Volpe. Mindscape pulls the server plug on nabaztag, hands source code to developers. *http://www.engadget.com/2011/07/28/mindscape-pulls-the-server-plug-on-nabaztag-hands-source-code-t/*, 2011.

[18] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a wireless sensor network on an active volcano. *Internet Computing, IEEE*, 10(2):18–25, March 2006.

[19] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. Johnson. M2m: From mobile to embedded internet. *Communications Magazine, IEEE*, 49(4):36–43, April 2011.