Dynamic Relay Node Placement in Wireless Sensor Networks

Jorge Mena, Vana Kalogeraki
Department of Computer Science and Engineering
University of California, Riverside
Riverside, CA 92521
{jmena,vana}@cs.ucr.edu

Abstract

In this paper we present an online algorithm that attacks the problem of placing relay nodes in regions where high localized congestion is detected. Congestion refers to the network overload caused by excess concurrent attempts of wireless devices to access the common shared channel. Our algorithm uses the Expected Transmission Delay (ETD) metric based on both current measurements and history observations over a window of time and is used by the nodes to identify the level of congestion in the local environment of the node. ur algorithms use a dissemination probing mechanism to exchange statistical local information periodically which are used to detect the congested area. Our detailed simulation results illustrate the working and benefits of our approach.

1 Introduction

Event-based distributed sensor systems have attracted a lot of interest in the research community to track moving objects and monitor events of interest in real-time. Examples include emergency response systems, medical systems, command and control systems, military and surveillance applications. For example, in a battlefield surveillance scenario, a network of wireless sensor devices with sensing and radio communication capabilities can be deployed to inspect a geographical zone and identify potential threats in the territory. In a retail store or warehouse environment, RFID devices can be deployed for efficient supply chain management. Sensors or RFID readers can detect certain events, identify alarms and forward the information to a central server or to some business application.

However, an increasing usage of wireless communication devices has increased the demand for the communication resource that they all share: the *radio spectrum*. Unfortunately, given the limited spectrum resource and the increasing demand for the radio spectrum by the users, there is an urgent need to improve the spectrum allocation and use. In the United States, the Federal Communications Commission (FCC) has already identified this problem and taken action by recently releasing to the public some portions of the restricted radio spectrum as a measure to allocate more resources. However, this measure is still not enough to efficiently satisfy the demand. Furthermore, the unique characteristics of the sensor nodes, including the small queue sizes, the short batter life and the limited computational power, make the problem more challenging. In a sensor network, the sink node is the single destination of all the traffic generated at the sensor field. The initiation of a large number of simultaneous flows is likely to cause congestion, first, due to a large demand of bandwidth close to the sink, and, second, due to collisions of packets from simultaneous flows traversing interfering paths.

Current approaches that address congestion focus on reducing the rate of the flows that enter the congested region, such as [6], or reroute traffic through virtual relay node sinks rooted at the actual sink [14]. Relay node placement approaches currently place relay nodes a priory without considering the network traffic load or congestion occurring at run time and they rather concentrate on problems such as maximal area coverage [9] or look for a fault-tolerant sensor network [16].

The goal of this paper is to provide an on-demand relay placement mechanism to react to a congested region and reroute traffic to temporarily mitigate the problem. Congestion refers to the network overload caused by excess concurrent attempts of wireless devices to access the common shared channel. Congestion can be caused by several factors such as resource unavailability (channels, queues, etc.), due to high load or hardware failures. Our idea is simple, *we want to add additional resources to a congested region in the network in order to reroute the traffic and alleviate congestion.* We provide a distributed dissemination probing mechanism that exchanges statistics locally in the network that are used to determine the placement of the *relay nodes* in the region. A relay node is a powerful node equipped

with several wireless interfaces[1] and able to tune to distinct wireless frequencies that do not interfere with one another. In the sensor network, we use the relays to eliminate congestion by creating additional paths to reroute traffic. Relay nodes self-organize into a separate network, changing the topology of the network and forming links with sensor nodes, to minimize congestion and improve network performance. Our goal is to provide a fast solution to the problem while a more permanent solution can be implemented using additional hardware. Therefore, our work is intended for sensor networks currently deployed in an application setting. We do not make many assumptions, but we assume that the devices are organized in an ad hoc network and use the 802.11 protocol to operate. Our experimental results illustrate the working and benefits of our approach.

2 Related Work

The literature has several approaches on how to use the channels and interfaces in a wireless network. Wan in [14] utilizes virtual relay nodes to attack the "funneling effect" generated at the sink. With an in-band signaling system encoded onto the packets, sensors may discover new routes through the relays rooted at the sink. If the sink is equipped with several interfaces, then the created overlay may become completely independent of the existing network, thus eliminating interference. We have a similar dynamic behavior with the difference that we do not modify any of the packets from the underlying protocols that could make the entire system non-standard. We prefer to adopt an improved routing algorithm rather than changing the underlying packet structure. Zhang and Misra [16] study four related fault-tolerant relay node placement problems and offer a constant polynomial time complexity. Their work is based on Steiner minimum trees where they try to minimize the number of relay nodes into a fixed location placement. Contrary to our work where we assume a dynamic environment where the network traffic varies with time, we found that this approach is not appropriate for our setting. Instead, we focus on covering the regions with high detected localized congestion, rather than minimally cover the entire network. Lloyd and Xue propose relay node coverage at the initial setup of the network where all the nodes are required to have a path to all the relay nodes in the overlay network [9]. They also use Steiner Trees to minimize the number of relay nodes. We adopt their method of placing the relay nodes so that they are evenly placed on a straight line that connects two points, but we perform this fix dynamically, as opposed as in the initial setup. Wang, et al, investigate in [15] the problem of self-organizing topologies to uniformly maximize the coverage area of a sensor network. Their method of selecting a new location is similar to ours, except that we need to address extra constraints when we place a relay node close to a congested node. Srinivasan, et al, in [10] address the problem of identifying the contour of a region characterized by the sensed activity. We determine the contour of our congested region by calculating the convex hull of the set of congested nodes.

Florea and Yanikomeroglu argue in favor of the use of relays in a cellular network in [2]; they conclude that relay nodes are an essential component to handle congestion in a network but the number of relays must be minimized to achieve efficiency. Although their work is based on the cellular network system, their conclusions can also be applied to sensor networks. We do not concentrate particularly on placing the minimum number of relay nodes in our experiments, but rather we try to minimize the expected end-to-end delay of our traffic flows; however, we do not arbitrarily place our relay nodes on the network. Although we do not guarantee minimum number of relay nodes, we show in our results that we utilize a considerably smaller amount of relays than random placement.

Gupta, et al., make a pioneering capacity analysis on a single-channel network in [3] and their work is further extended by others in [8, 7, 11, 1, 12, 5, 4]. We base our capacity analysis on their observations. Wan, et al, provide a classical analysis and technique of congestion detection and avoidance called CODA in [13]. They provide a channel and congestion analysis to detect congestion. In our previous work [6], we provide an analysis of congestion based on a queuing model. In this work we focus on the relay node placement problem.

3 Network Model

We consider a wireless sensor network with n nodes, located in a 2-dimensional plane. We represent the network as an undirected graph $G(V, E)$, where V is the set of vertices, or the nodes of our network, and E is the set of edges, or communication links. Each node is characterized by its transmission range, r. We associate a set of channels C_u to each node u, where C_u is a subset of C, the set of channels that the wireless nodes can listen on. Assuming that the geographical location of two vertices in the graph, u and v on the 2-dimensional plane, then the Euclidean distance that separates the two vertices in the graph, u and v (or nodes in the network), is $d(u, v)$. An edge $e = (u, v)$ belongs to the set E if the following is true: (1) $d(u, v) < r$, (2) $C_u \bigcap C_v$. In other words, two wireless nodes can communicate if they are within their transmission range and they have a common channel to use.

A node u is able to tune its interfaces to any one of its channels C_u to generate a transmission signal to a neighboring node tuned on the same channel. We assume that a node

[1] We use the terms *radio* and *wireless interface* interchangeably.

is either statically assigned channels to its interface, or algorithms can perform the channel assignment. Furthermore, we make no assumption about how routing paths are established in the sensor network. Our metric is orthogonal to the routing protocol and can be used with many popular routing protocols such as DSR and AODV. With today's commodity wireless cards, the time required to re-tune an interface is on the order of tens of milliseconds; thus, a high demand of channel retuning may introduce unnecessary complexity to control a potential inefficient channel-switching behavior. In our work we assume a wireless sensor network with an initial setting, in which our aim is to identify the congestion hot spots and bottlenecks, and then divert the traffic, on-demand, to relieve these areas.

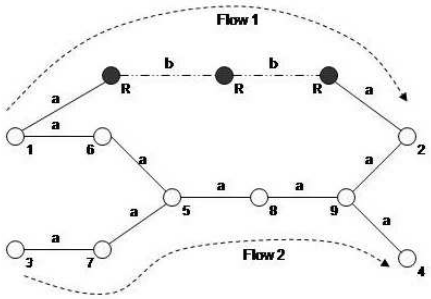


Figure 1. Wireless sensor network with relays.

3.1 Relay Nodes

The relay nodes are multi-interface, multi-channel wireless nodes, organized into a separate network, whose goal is to create an alternative path across a congested geographical area. Since relays are multi-channel nodes, they may tune to the currently congested channel to pick up traffic and transport it with other relay nodes using an available non-congested orthogonal channel. This way, the relay node network will introduce new routing paths to a congested region in an attempt to "patch" the network or "detour" the traffic and thus eliminate congestion. In Figure 1 we show how we can create a detour for the traffic flows using a network of relays. In this multi-channel network we illustrate two single-channel networks: one composed of the relay nodes using channel b and another one composed of all the sensor nodes using their common channel a. This method does not require any change of the underlying network because the relay nodes may tune one of their interfaces to the channel that the network uses to pick up some traffic that they forward among the relay nodes themselves using another non-interfering (orthogonal) channel. This approach effectively reduces congestion. Moreover, it adds more resources to an existing network.

4 Metrics

We define the following metrics to evaluate the performance of our approach.

Expected Transmission Delay (ETD). The ETD metric represents the time required to transmit a packet to a neighbor hop. The metric is calculated as follows: Every time a node receives a packet, be it a probe, data or control packet, it measures the hop delay D_{e_c} that this packet experienced to get to this node. Notice that this delay takes all the retransmissions, channel contentions, and queue service delays into consideration[2]. This delay is stored on a per-link basis. Then, we compute the expected transmission delay ETD as the weighted average of the past ETD computations, within a sliding window w and the current packet delay recorded, as follows:

$$\text{ETD}_{e_c}(t) = \alpha * D_{e_c} + (\alpha - 1) * \text{ETD}_{e_c}(t-1) \qquad (1)$$

This metric allows us to use both the past observations and the current delay measurements at the node. In [1], De Couto argues that current delay observation is not enough to identify a path as better than another one because instability may cause oscillations in the decision. With this in mind, we use the tunable parameter α. An α value closer to 1, puts greater weight on recent delay measures; while an α value closer to zero, puts more weight on the past measurements.

Path Cost. We can then compute the transmission cost of a path P between two nodes u and v as the sum of the expected transmission delays of each link that constitute the path, as follows:

$$PC_{P_{uv}} = \sum_{e_c \in P_{uv}} \text{ETD}_{e_c}, \forall c \in C \qquad (2)$$

where e_c is an edge in $G(V, E)$ or a hop in the path P. Given a path, the transmission cost of the path is dominated by the link with the largest delay. Therefore, one path is determined to be better than another one by a simple comparison of their values.

Drop Count. Our nodes also keep a count of the number of dropped packets $dropped\_packets_{e_c}$ observed over a sliding window time w. We use the drop count statistic as a supporting indication when identifying a congested link. Usually, when congestion occurs, a ripple effect occurs in the surrounding region where the congested link is detected. A node that forwards packets might face a high contention of a channel or due to a hidden-node problem. Subsequently, the node may experience an abnormal increase of its queue size, which eventually causes it to overflow, resulting in packet drops.

[2] Our metric assumes that the nodes are time synchronized; for example, they can use the GPS network to synchronize their clocks.

5 Relay Node Placement

Our proposed mechanism aims (1) to identify regions of high localized congestion in a wireless network and (2) to determine the geographical placement of relay nodes in the 2-dimensional planes in order to mitigate congestion. The overlay network of relay nodes is capable of using the underutilized orthogonal channels in the congested region with the use of its several radio interfaces capable of establishing full-duplex paths. These paths are the new resources that we try to add to the underlying congested network to introduce new routing paths formed of non-interfering channels, in an attempt to reduce some of the traffic of the underlying congested region. We work under the assumption that there exists a limited number of orthogonal channels that any relay or sensor node may tune to at any time. Also, we assume that we have a large number of relay nodes at our disposal; although we do not impose a limit on the number of relays that we have available, we try to use the smallest number of relay nodes possible. When the locations are determined, we assume that a human or a robot can place the nodes in location. We describe this process below:

5.1 Identifying Congestion

The placement of relay nodes starts with the detection of the congested region that will be "patched"; this task is performed by a $head$ node that collects the region statistics and executes the relay node placement algorithm. When a node finds that it is congested, it assumes the role of the $head$ node of its region. It initiates a message exchange protocol during which adjacent congested nodes merge their regions to form one larger congested region with a single $head$ node[3]. The idea is to determine the congested region so that we can introduce new resources in the form of new route paths, thus "patching" the network with relay nodes. Once we determine this region, the $head$ node proceeds to patch it. A node can detect that it is congested when:

- There is a large variance in the ETD metric observed when transmitting packets to a neighboring node. Instability in the transmission delays is a clear indication that congestion occurs. Under high loads, the channel will be constantly busy, causing large delays for packets in the queue that will be later dropped. Suddenly, the congested node becomes resource available and its acceptance packet latency suddenly drops until the queue fills up again. This unstable behavior is therefore a clear indication of packet drops and high demand of the channel.
- There exist drops caused either by packet collisions or queue overflows. The drop count keeps track of the number of packets dropped within a sliding window w. A high drop count value indicates a high load in the congested area.

When a node detects that it is congested, it broadcasts a congestion message to all its neighbors. Each neighbor will determine if it is also congested, an if so, it will reply. If the neighbor is congested, the congested regions will merge and they will repeat the same process with their own neighbor nodes. If the neighbor node is not congested, this node becomes a "frontier" node, respond to the message, and stop the dissemination of more congestion messages as they have found the edge of the congestion region.

5.2 Placing Relay Nodes

Algorithm 1 Relay Node Placement

Input: Region of congested nodes C
Output: None

- Compute the Convex Hull of the congested nodes set C
- For each frontier node in the Convex Hull :
  - Find the non-congested neighbor nodes of the frontier node that introduce traffic to the congested region
  - Use the $CoverRegion$ algorithm to cover with relay nodes the region composed of the frontier node and all its neighbors. Let this set of relays be R
- Compute the clusters of the set R using K-means with a value $k = size(ConvexHull)$. Let this set of relay clusters be CL
- While the size of $CL > 1$:
  - Select cluster with the smallest size, $cl1$
  - Select the closest cluster $cl2$ to $cl1$
  - Join these two clusters with the $CoverWithRelays$ algorithm by adding relay nodes on the straight line that connects the closest two relay nodes in these two clusters

Once a congested region CR is detected, our aim is to place relay nodes at a minimum distance of $r + \Delta$ from any congested node n_b that belongs to CR to guarantee that the new relay nodes will not further contribute to congestion.

[3]In the case that multiple nodes experience congestion, thus there are multiple head nodes, a single $head$ node can be elected, e.g. the one with the smallest node id.

By definition, a *frontier node* is one that has a congested node as a neighbor; however, a neighbor of a frontier node might not be congested; these are called *entry nodes*, see Figure 2(c). We aim to place relay nodes at locations surrounding the entry nodes of a congested region that forward traffic flows that contribute to congestion.

Also note that at the edges of the congested region, the entry and frontier nodes naturally form clusters. Relay nodes that connect with the underlying network must cover the entry nodes, thus also forming clusters. Therefore, to cover a congested region, the relay nodes must cover the entry nodes and form clusters in a first step; once all the relay clusters are formed, in a second step, all these clusters are linked together to form a single connected overlay network and thus providing the new relay routing paths. Next we explain how these two steps are performed. The high level description of the relay node placement algorithm is shown in Figures 2 and 3.

The Convex Hull and the Node Clusters. Before relay nodes are placed, the head node of the CR needs to identify the geographic locations of the frontier nodes. For this, the Convex Hull of the region is calculated. This will return a set of nodes with mostly all the frontier nodes and eliminate the internal congested nodes of the region. It may be possible that some congested nodes are part of the convex hull but they are ignored.

From Figure 2(a), the congested nodes neighbor with a frontier nodes and these neighbor with entry nodes (non-congested neighbors). Both frontier and entry nodes form a local cluster at one entry point of the congested region, but relay nodes are placed only within the transmission range of the entry nodes, thus forming local clusters of relay nodes for the congested region.

Once the entry nodes are identified at each local cluster, we proceed to cover them with relay nodes using the routine $CoverRegion$. At the end of this routine, relay nodes are placed and a set of relays R is returned and added to the already existing set Z. At the end, all local clusters are covered with relays ready to be joined to form the overlay network.

Joining the Relay Clusters. After relays are properly placed to cover the entry nodes, we need now to identify the local relay clusters and join them. The head node takes the set Z and runs the **K-means** algorithm with a k value equal to the size of the Convex Hull set calculated in the previous set. K-means is used because it uses the euclidean distance to form the clusters and identify the members of each cluster.

With all the relay clusters identified, the head node proceeds to join them one by one with the $CoverWithRelays$ routine. This routine takes two clusters and draws a line between the two relays (one per cluster) that are the closest and covers it with equally distant relay nodes. The result is two clusters joined into one single connected-component. This process continues until only one network is formed. Figure 2(b) shows how two clusters are joined together.

CoverRegion Routine. This routine places the necessary relay nodes that cover a region denoted as the input set S of nodes. The set S is composed of the congested nodes (usually one) n_b, the frontier node, and the entry nodes of the cluster. A relay node n_r is placed at a location closest to the frontier node and within the transmission range of the entry nodes such that the following condition is not violated:

$$d(n_r, n_b) \geq r + \Delta$$

The condition above is applied to prevent the use of the congested channel by any relay node when it attempts to communicate with the underlying network.

The first step is to find the extrema points of the region. These two points are the two farthest points in the area computed by the intersection of the transmission range of both the frontier n_f and the congested n_b nodes. The extrema points are defined as follows:

- $dist(p', n_b) = r, \quad c \in C$
- $dist(p'', n_f) = r, \quad c \in C$
- $p' = p''$

If there are more than two extrema points, we select the farthest two.

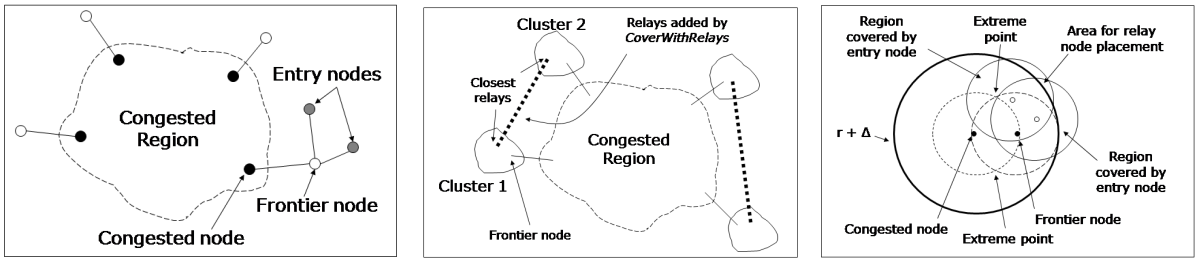
Each extreme point is used as the starting reference location where a relay node is placed. The reasoning behind the extreme points is the following: the extreme point is the closest point that an entry node can be located such that it is still not within the transmission range of the congested node (otherwise the entry node would be a frontier node). For this reason, entry nodes are on the extreme point or away from the congestion region, see Figure 2(c).

Once the extrema points are identified, the routine $Cover$ is called to cover the entry nodes at each of these two points. With this, we guarantee that all the entry nodes that are closer to the congested node are covered; this is important because we want to place relay nodes the farthest away from the congested node.

After the extreme points are covered, there may be other entry nodes that are not covered yet; however, they are simply covered by calling the $Cover$ routine on their locations. This step is simpler since the remaining entry nodes are the farthest located from the congestion node.

At the end, a new set of relay nodes R is returned covering the region denoted by the input set S. The algorithm for the CoverRegion in shown below:

$Cover$ **routine.** This routine determines the actual placement of the relay nodes. A node n is taken as input to



(a) The Convex Hull (b) CoverWithRelays routine (c) Area to cover with relays

Figure 2. Illustration of our approach.

Algorithm 2 CoverRegion

Input: Region of nodes S
Output: Set of relay nodes R that cover the region spanned by nodes S

- Find the two extrema points of the set S
- Use the $Cover$ algorithm to cover the closest node to any of the extreme points
- Use the $Cover$ algorithm to cover the 2nd closest node to the other extreme point
- For any remaining node uncovered in the set S :
  - Use the $Cover$ algorithm to cover this node
- Connect all the newly added relays R to form one single connected component

be covered. First, we find those nodes neighbors of n that introduce traffic into the congested region; these nodes form the set N sorted in order of proximity to n. We then proceed as follows: we find the *closest* point p to n such that p is outside the interference range of the congested node n_b. Then, for each neighbor node n_i in N we find the *closest* point q such that q still covers all the nodes covered by p. This stops until q does not cover all nodes previously covered by p. We place a relay node at p and the routine ends. Figure 3 shows the working of our approach. From this figure we see how the potential point p moves away from the extreme point and covering all closest neighbors of the input node n until they are all covered or until the new p starts uncovering nodes.

$CoverWithRelays$ **routine.** This routine is very simple. It takes two points, namely $p1$ and $p2$. It then calculates the inter-relay node distance along the line ($p1$, $p2$) and it places a relay node there. It lastly returns the set S with relay nodes that connect these two points.

Algorithm 3 Cover

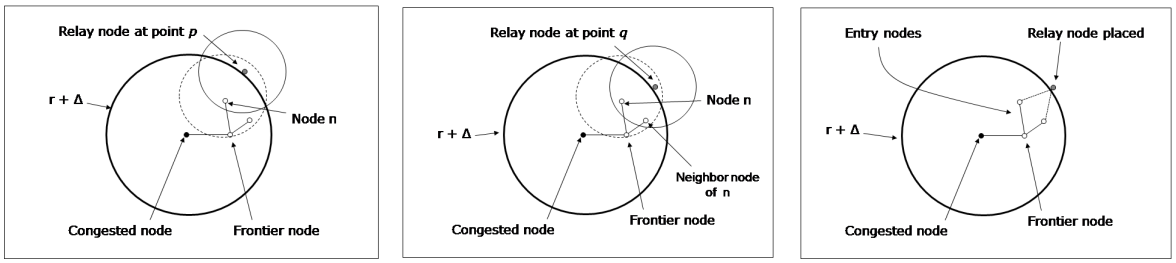
Input: Node n, set of uncovered nodes U, congested nodes C, relay set R
Output: Return R

- Find the neighbor nodes N of n that introduce traffic into the congested region C
- Sort N in order of proximity to the node n
- Find the closest point p inside the transmission range r of n such that the distance from any congested node to $p > r + \Delta$
- For all the nodes n_i in N :
  - Find the closest point q to n_i such that the distance from any congested node to $q > r + \Delta$
  - If q does not cover all the nodes covered by p, break this loop
  - Let $p = q$ // p is the current valid placement
- Place a relay node at the location p and add it to the set of relays R

Algorithm 4 CoverWithRelays

Input: End points $p1$, $p2$ and set of relay nodes R
Output: Minimal set of relay nodes S that evenly cover the line ($p1$,$p2$).

- Calculate the largest inter-relay spacing sp between the points $p1$ and $p2$ such that $sp < r$
- Cover the straight line ($p1$,$p2$) with relays such that for each needed relay position p, if there exists a relay n_r in R, use n_r instead of adding a new relay; otherwise, add a new relay located at p into S



(a) First placement of p (b) A new point q that still covers p (c) Final placement of a relay node at p

Figure 3. The $Cover$ routine

6 Evaluation

We have implemented and tested our approach using the the Network Simulator ns2, to implement the probing mechanism, congestion detection, and node placement algorithms. We used a sensor network of 25 nodes that use a single channel for communication. The nodes are configured to use the 802.11b protocol with a maximum data rate of 2Mbps and control rate of 1Mbps and a transmission range of 225m. We use the default channel of 2.472GHz (channel 13) of the spectrum. Relay channels operate on two other orthogonal channels in the range of 2.412GHz (channel 1) and 2.437GHz (channel 6), separated at least by 30MHz to make them non-overlapping.

In our traffic simulation, we observe the network behavior in periods of 100 seconds; we inject traffic flows with a rate of 10 packets per second of size 1000 kilobytes (plus the packet header bytes of the network protocols). These correspond to a sudden congestion (such as the rush hour in a cellular network or a fire in a forest covered with sensor nodes).

Probing. In the first experiment we evaluate our probing mechanism. In Figure 4(a) we show the results of using different w values; we concluded that a period of 1 second had the best balance between channel load measurement accuracy and bandwidth consumption. Each node generates and broadcasts a probe packet of 64 bytes that contains the source node id and a time stamp to calculate the hop delay among peers. The neighbors, in response, reply back to the source to allow the original sender to calculate its reverse delay. All nodes calculate their local statistics in this way and keep a history of $\tau = 10$s. We keep a value of $\alpha = 0.5$, which means that we equally weight the observed average delay and the ETD calculated with the expected forwarding and reverse delays; this way, we equally weight the historical observations with the current measurements of delay.

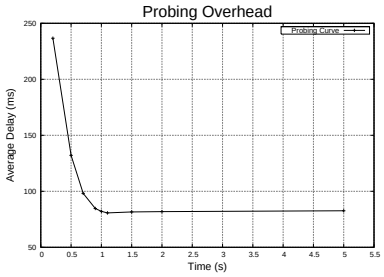
Congestion Detection. We run several experiments on several topologies with a common bottleneck connecting two or more regions; however, due to the lack of space we chose Figure 8(b) to present our results. The network in Figure 8(b) is composed of three clusters of sensors connected by a "Y" shaped bottleneck path. We introduced three flows that go through the bottleneck to reach their destinations, thus purposely creating congestion. The flows are the following:

- Flow 1: Node 8 $\rightarrow$ node 24
- Flow 2: Node 21 $\rightarrow$ node 15
- Flow 3: Node 18 $\rightarrow$ node 10

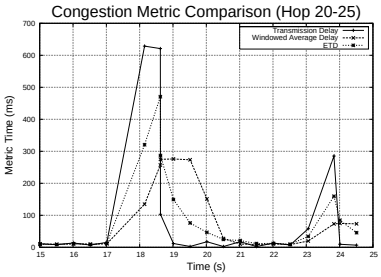
With the three sources injecting packets onto the network, nodes in the bottleneck start experiencing delays delivering their packets creating a ripple effect back to the flow sources. Figure 4(c) evaluates the accuracy of the ETD metric by comparing ETD with the average delay and the current transmission delay, when detecting congestion. The figure shows a sudden increase in the drops recorded by node 20 (the entry point to the bottleneck) at an approximate time $t = 18$s. At this point, node 20 checks its statistics to find out if it is congested by calculating the standard deviation of its ETD. We observed that a standard deviation of twice the window average is a good indication of a widely spread ETDs, which indicates the presence of instability in the network (periods of long delays followed short ones that indicate a cleared/overloaded queue). At this point, we say that node 20 is congested.

In Figure 4(b) we show that the solely use of the window average is not a good indicator of the status of the network. For example, in the period between $t = 18$s to $t = 21$s we observe how the average delay falsely detects prolonged delay period when in fact it is an instability period. A similar situation happens at $t = 23$s. ETD does not suffer of this problem and with the use of standard deviation to investigate the severity of the instability, we determine if a node is congested or not.

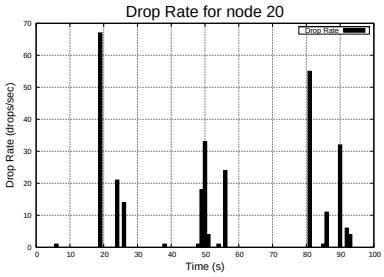
Placement of Relay Nodes. In the next set of experiments we evaluate the placement algorithm. When a node detects that it is congested, it starts to investigate the extend



(a) Probing Overhead

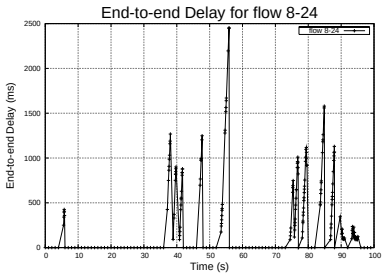


(b) Metrics comparison at hop 20-25

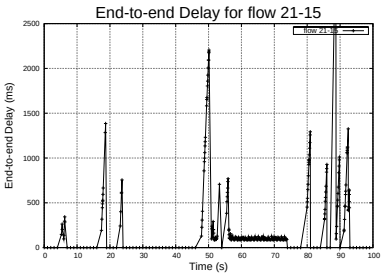


(c) Drop Count at node 20

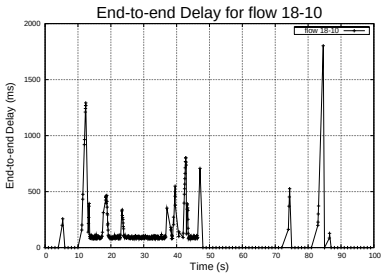
Figure 4. Congestion Detection.



(a) Congestion on flow 8-24

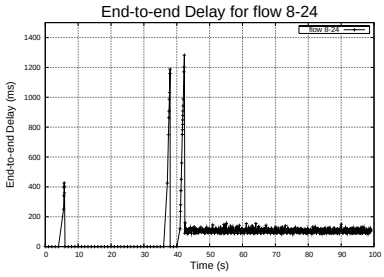


(b) Congestion on flow 21-15

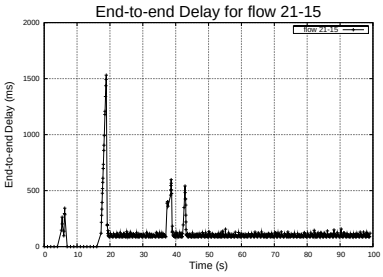


(c) Congestion on flow 18-10

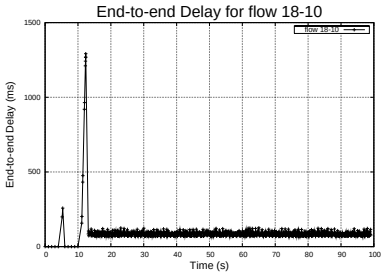
Figure 5. End-to-end delays of the network without relays.



(a) Flow 8-24 re-routed through overlay network



(b) Flow 21-15 on underlying network



(c) Flow 18-10 re-routed through overlay network

Figure 6. End-to-end delays of the flows with the presence of relays.

of the congested region. From our experiments we observed that the time that the congested node took to determine the extent of the congested region is proportional to the size of the congested region. For example, the largest congestion region is the one of node 6 and it took approximately 1295ms to be calculated (node 20 took 1115ms and node 13 took 46ms). We assume that congestion messages are given the highest priority and that the processing time to calculate if a node is congested is negligible (table look-ups and standard deviation calculation).

In Figure 5(a) we show the end-to-end delays of each of the flows in the network without relays. The high spikes in each figure represent clear periods of instability in the network and similar conclusions can be derived from Figures 5(b) and 5(c). Also note the periods of zero end to end delivery at each of these three figures. In Figure 5(a), the zero delivery at $t = 10$s to $t = 35$s is caused by the dominant flow 3 in the same period in Figure 5(c). A similar reasoning can be derived about the other two figures. As a result, we observe that only flows 2 and 3 dominate the entire simulation time.

We also report the corresponding end-to-end delays of the flows after the introduction of the relays. The new position of the relay nodes is observed in Figure 8(c), in which we can observe that the overlay network connects to the entry points of all the three congestion regions. Also note that links among relays do not affect the underlying network because they use a distinct channel (these links are shown as dotted). Flow 1 enters the overlay at relay R2 and leaves at relay R1. Flow 3 also uses the overlay via R6 and R3. Flow 2 remained on the underlying network. The results of the end to end delivery of these three flows can be observed in Figures 6(a), 6(b) and 6(c). Once congestion is detected (indicated in high spikes), the relay nodes pick up the traffic and thus reroute the traffic to eliminate congestion. This shows in the figure with the decrease in the end-to-end delays.

We also calculated the throughput of each flow measured in packets delivered per second in Figures 7(a), 7(b), and 7(c). Remember that each flow is injecting 10 packets per second, so this value is our upper limit and that congestion is injected at time 5s. From the three graphs we can observe the increase in packet delivery before and after the user of relays and rerouting. It interesting to note the high throughput delivery at times when the flow is dominating and relays were not being used. For example, in Figure 7(c) we can observe the high throughput delivery at $t = 25$s which coincides with Figure 5(c) at the same time. When relays were used, this is also corroborated.

Lastly, we compared the performance of our approach to a random placement allocation (given the same calculated congestion region) in terms of number of relays used. In the random placement you randomly select the location of the relays in the congested region and you stop this process when all entry nodes are connected. Our results are given in a the histogram in Figure 8(a). In all three cases we beat the random allocation by about four times the number of relays that we used on average. We do not focus on minimizing the number of relays used, but rather on the congestion detection to trigger the algorithm; however, we observe that we significantly reduced the number of relay nodes used to achieve similar delivery delay results.

7 Conclusions and Future Work

This work provides a solution to the relay node placement in a network of wireless sensor devices. Our algorithm places a number of relays in the congested region with the constraint that no relay node that communicates with the underlying network is placed within a distance of $r + \Delta$, to avoid the introduction of more traffic in an area already congested. We use the ETD metric to identify a congestion region. This work can be utilized in situations where immediate action is required, such as emergency response scenarios where a high rate of packets are generated. There are many avenues for future work. One important line of research is the channel assignment problem. Relays must choose what channels they must tune to in the deployment phase. Another future improvement is the incorporation of distinct statistical analysis tools for the detection of congestion.

References

[1] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing, 2003.

[2] A. Florea and H. Yanikomeroglu. On the scalability of relay based wireless networks. In *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, 2006.

[3] P. Gupta and P. Kumar. Capacity of wireless networks, 1999.

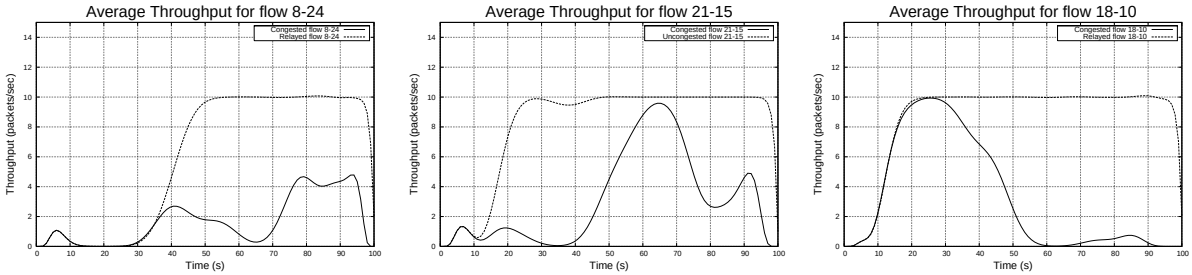
[4] A. P. Jardosh, K. N. Ramachandran, K. C. Almeroth, and E. M. Belding-Royer. Understanding congestion in ieee 802.11b wireless networks. In *IMC'05: Proceedings of the Internet Measurement Conference 2005 on Internet Measurement Conference*, pages 25–25, Berkeley, CA, USA, 2005. USENIX Association.

[5] J. Jun, P. Peddabachagari, and M. Sichitiu. Theoretical maximum throughput in ieee 802.11 and its applications.

[6] K. Karenos, V. Kalogeraki, and S. V. Krishnamurthy. Cluster-based congestion control for supporting multiple classes of traffic in sensor networks. In *RTSS*, 2005.

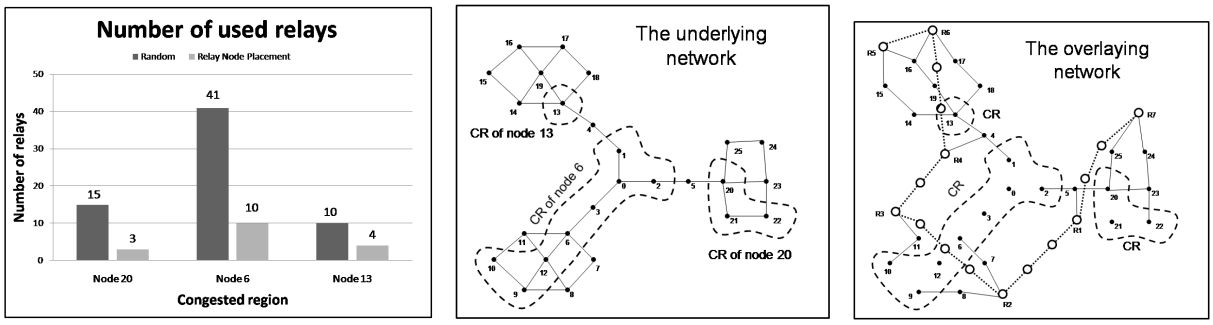
[7] P. Kyasanur and N. H. Vaidya. Capacity of multi-channel wireless networks: impact of number of channels and interfaces. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 43–57, New York, NY, USA, 2005. ACM Press.

[8] J. Li, C. Blake, D. S. D. Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *MobiCom '01: Proceedings of the 7th annual international conference on*



(a) Throughput of flow 8-24 (b) Throughput of flow 21-15 (c) Throughput of flow 18-10

Figure 7. Throughput observations per flow in packets per seconds with and without relays.



(a) Number of relay nodes used (b) The Network without relays (c) The Network with relays

Figure 8. Topology of the network with and without relays.

Mobile computing and networking, pages 61–69, New York, NY, USA, 2001. ACM Press.

[9] E. L. Lloyd and G. Xue. Relay node placement in wireless sensor networks. *IEEE Transactions on Computers*, 56(1):134–138, 2007.

[10] S. Srinivasan, K. Ramamritham, and P. Ramanathan. Contour estimation using collaborating mobile sensors. *International Conference on Mobile Computing and Networking*, 2006.

[11] J. Tang, G. Xue, and W. Zhang. Interference-aware topology control and qos routing in multi-channel wireless mesh networks. In *MobiHoc '05: Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 68–77, New York, NY, USA, 2005. ACM Press.

[12] S. Toumpis and A. Goldsmith. Capacity regions for wireless adhoc networks, 2001.

[13] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell. Coda: congestion detection and avoidance in sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 266–279, New York, NY, USA, 2003. ACM Press.

[14] C.-Y. Wan, S. B. Eisenman, A. T. Campbell, and J. Crowcroft. Siphon: overload traffic management using multi-radio virtual sinks in sensor networks. In *SenSys '05: Proceedings of the 3rd international conference on Embedded networked sensor systems*, pages 116–129, New York, NY, USA, 2005. ACM Press.

[15] G. Wang, G. Cao, and T. L. Porta. Movement-assisted sensor deployment. *IEEE Transactions on Mobile Computing*, 2006.

[16] W. Zhang, G. Xue, and S. Misra. Fault-tolerant relay node placement in wireless sensor networks: Problems and algorithms, 2005.