DEAL: Discover and Exploit Asymmetric Links in Dense Wireless Sensor Networks

Bin Bin Chen*, Shuai Hao†, Mingze Zhang*, Mun Choon Chan* and A.L. Ananda*
*School of Computing, National University of Singapore. {*chenbinb,zhangmi3,chanmc,ananda*}*@comp.nus.edu.sg*
†Computer Science Department, University of Southern California. *shuaihao@usc.edu*

[1] ***Abstract*— Asymmetric links commonly exist in low power wireless sensor networks. However, it is difficult to discover and exploit them efficiently. In this work, we propose DEAL, a link management scheme to Discover and Exploit Asymmetric Links efficiently in dense wireless sensor networks. Equipped with a novel feedback mechanism, DEAL dynamically adapts its link maintenance mechanism based on the estimated link quality, and manages the (small) neighbor table so as to retain the most useful information.**

We implement DEAL in TinyOS and evaluate its performance using both TOSSIM and testbed. The simulation results show that more than 80% of asymmetric links can be discovered and maintained with minimum overhead. Using a collection tree application and ETX as the routing metric, the average path ETX can be reduced by up to 20%. Testbed evaluation also shows that DEAL improves the network routing performance by identifying useful asymmetric links.

I. Introduction

With low-power wireless nodes, asymmetric links are common even if the same transmit power is used. Measurement studies in [1], [2], [3], [4] have demonstrated the presence of asymmetric links. Amount of asymmetric links increases with lower power [1]. The fraction of asymmetric links is high enough that topology control mechanisms should take such links into account [2]. Many reasons contribute to the existence of asymmetric links, including transmission power disparity, interference, obstacles, noise level difference, as well as radio irregularity [5].

By considering both symmetric and asymmetric links, network connectivity improves. In addition, measurement results from our testbeds (as well as the work in [1]) have shown that asymmetric links tend to span longer distance than symmetric links. As a result, inclusion of asymmetric links can improve network performance in terms of both routing reliability and efficiency.

While there are many benefits in considering asymmetric links, efficient discovery and exploitation of these links remain a challenge due to the following reasons: (1) Link discovery and exploitation heavily rely on the information exchanges between the neighbor nodes. Thus, efficient information delivery over the "poor" direction of the asymmetric link is the fundamental prerequisite to use it. (2) The time-varying link quality requires the link maintenance strategy to provide both accurate and efficient measurement. (3) The hardware limitation of cheap sensor nodes often places additional barrier on asymmetric link exploitation.

In this paper, we present *DEAL*, a link management scheme to Discover and Exploit Asymmetric Links efficiently in dense wireless sensor networks. We make the following contributions in this work:

1) We propose *Source-Specified Relay (SSR)*, a novel information feedback scheme over the poor direction of the asymmetric links. SSR is much more efficient than existing information feedback schemes [6], [4], which incur either high communication cost or high memory consumption. For SSR, both the memory and communication costs are constants $O(1)$.

2) To maintain and exploit asymmetric links under different link quality situations, we propose *Dynamic Driven Maintenance (DDM)*, a link maintenance scheme which dynamically adopts different strategies so the most efficient link maintenance scheme will be utilized depending on the link quality.

3) We design the *Asymmetry-Aware Caching (AAC)* neighbor table management scheme, which systematically addresses the impact of the limited memory size on the asymmetric link maintenance. To our knowledge, this is the first work to consider this. All the existing works [1], [2], [3], [4], [7], [8] simply assume that the memory size is unlimited.

We have implemented DEAL in TinyOS, and study its impact on a tree collection application. Simulation using TOSSIM shows that more than 80% of asymmetric links are discovered and maintained with minimum overhead, which improves routing layer packet delivery efficiency by up to 20%. Testbed evaluation also shows that the proposed schemes indeed exploits the asymmetric links which enrich the network connectivity and improve routing efficiency.

The paper is organized as follow. Section II discusses asymmetric link measurements from our sensor testbeds. Section III presents existing techniques to discover and exploit asymmetric links. Section IV describes SSR and DDM. Section V describes the design of AAC. Performance evaluation is presented in Section VI and we conclude in Section VII.

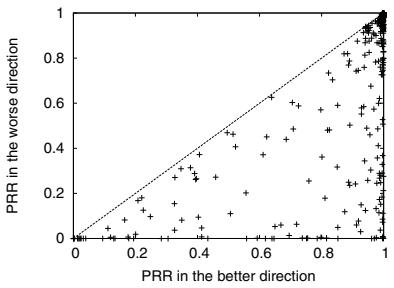
II. Characteristics of Asymmetric links

A. Testbed Measurements

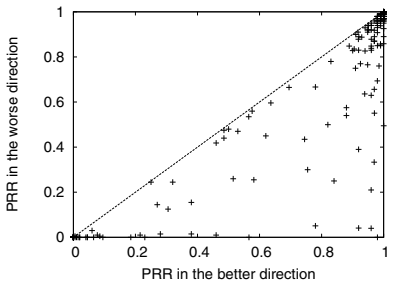
In this section, we motivate our problem by presenting link measurement results from 2 sensor testbeds deployed.

[1]The first three authors contributed equally to this work. This research is supported by National University of Singapore under grant R-252-000-265-112.

978-1-4244-2908-0/09/$25.00 ©2009 IEEE



(a) PRR for MICA2 (433MHz)



(b) PRR for MICAz (2.4GHz)

Fig. 1. PRR measurements on testbeds

The first testbed, consisting of 34 MICA2 (433MHz) motes, is deployed in an indoor office environment on a single floor (with walls and barriers). The distances among these nodes vary from 3m to 60m. In our measurement, all nodes use the same transmit power ($-10dBm$) and each node takes turns to broadcast packets. The packet reception ratios (PRR) are then recorded. The measurements are taken over five different 24-hour intervals. Over the entire measurement period, each node broadcast a total of around $25,000$ packets.

A scatter plot of the PRR is shown in Figure 1(a). Each point in Figure 1(a) represents a link between a pair of nodes. The horizontal axis shows the PRR in the better direction, and the vertical axis shows the PRR in the worse direction. All the points lie below the diagonal line, which represents the complete symmetric line. It can be seen that most of the points are located at the top right corner where PRR values are high in both directions. At the same time, there are also a large number of asymmetric links. For example, 17.6% of these links have differences of at least 0.25 in their PRRs in the two directions.

While not shown in the figure, this measurement using MICA2 also indicates that as the distance increases, the percentage of asymmetric links grows. On the average, asymmetric links are 75% longer than symmetric links. This property of MICA2 links indicates two potential benefits of asymmetric links. First, inclusion of asymmetric links can significantly improve the overall network connectivity. Next, as asymmetric links tend to be longer than symmetric links, by including these (usually longer) links in the network topology, some long routes (in hop count) can be avoided.

A similar set of PRR measurement is also carried out on a second testbed, which consists of 40 MICAz (2.4GHz) motes deployed over two floors of a building. The distances among these nodes vary from 5m to 80m. In this measurement, all nodes are battery powered and use the same transmit power ($0dBm$)[2].

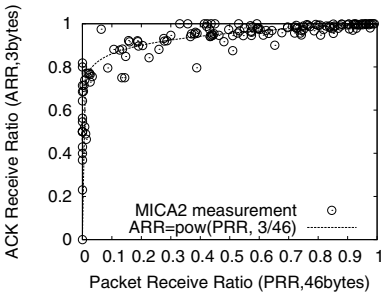


Fig. 2. ARR vs. PRR

The scatter plot of PRR pairs is shown in Figure 1(b). The percentage of asymmetric links is lower than the MICA2 testbed and the average distances between symmetric and asymmetric links follow similar trends as the MICA2 testbed. About 10% of these links have differences of at least 0.25 in their PRRs in the two directions.

It is interesting to note that our result is different from the measurement in [9] which shows that there are few asymmetric links using MICAz motes. We believe that there are two factors that contribute to the difference. First, the motes in [9] are deployed in an open area with little obstruction and motes are mostly in line-of-sight. Our measurement is based on an indoor environment with obstruction of walls. Second, the nodes in our testbed are powered by batteries, while the motes in [9] are AC-powered. As a result, the mote's transmission in our testbed can be affected by the battery voltage level.

B. ACK-capable Asymmetric Links

Due to the poor PRR in one direction, it is often assumed that asymmetric links cannot be used for packet transmission. However, this is not always true. For some applications, in particular, those involving data collection, reliable delivery of acknowledgement in the reverse direction is sufficient. Hence, in addition to PRR, ACK reception ratio (ARR) should also be measured. In this section, we will present results on why PRR is an overly pessimistic estimation of ARR and justify the use of certain asymmetric links.

There are two reasons why the ARR is higher than PRR over the same directed link. First, as also mentioned in [4], since ACK is sent immediately after the data packet, the channel around the sender tends to be clearer because of neighbor backoff. Second, MAC layer ACK is smaller in size. Take TinyOS 1.x radio stack as an example. For MICA2 motes (with CC1000 radio chip), packet size (S_{PKT}) can vary from 17 bytes to 46 bytes (including preamble and SYN bytes), while ACK size (S_{ACK}) is 3 bytes, and is sent immediately after data packet without any preamble sequence. In fact, since PRR is often estimated using large beacon packets (up to 46 bytes), the difference in packet size used for PRR and ARR is often very large.

[2]The transmit power are set independently in the two testbeds to create multi-hop network topology, taking into account of the radio characteristics.

Figure 2 plots ARR against PRR for all node pairs in our MICA2 testbed. The measurement is done by employing unicast communication between each pair of nodes. It is clear that ARR $\neq$ PRR and is in fact often much larger. For PRR as low as 10%, ARR can still be as high as 90%. From the figure, the independent loss model ARR $=$ PRR$^{S_{\text{ACK}}/S_{\text{PKT}}}$ [10] gives a good match to the measured data. When PRR is smaller than 10%, ARR can be any value in the range of $[\text{PRR}, 100\%]$. If the ARR in poor direction of an asymmetric link is higher than 70%, we call this asymmetric link an ACK-capable asymmetric link. In the MICA2 based testbed, out of all asymmetric links, around 70% of them are ACK-capable.

ACK-capable asymmetric links are useful because they provide high quality links for data forwarding and sufficient robustness for small acknowledgement packets. However, exploiting these asymmetric links requires additional tasks, which will be addressed in the following sections.

III. Existing Link Discovery and Exploitation Schemes

While asymmetric links can improve the network performance, discovery and exploitation of these links by both ends are complicated. Consider an asymmetric link between A and B (link quality from A to B is good), the information at B has to be fed back to A (on a poor link) before A can be aware of and possibly utilize the asymmetric link to B.

In this section, we will first briefly introduce the widely adopted Broadcast-based Active Probing (BAP) scheme [3], [11] and show why BAP cannot efficiently discover and exploit asymmetric links. Several existing mechanisms on how to resolve the drawbacks of BAP will then be described.

A. Broadcast-based Active Probing (BAP)

Broadcast is an efficient way for nodes to discover and maintain each other over symmetric links. In BAP, two nodes identify each other by exchanging "HELLO" and "I-HEAR-YOU" messages. When a node receives a "HELLO" message from another node, it identifies the latter as one of its inbound neighbors, and it will include the latter in the "I-HEAR-YOU" message and broadcast the message back to the latter. When a node receives "I-HEAR-YOU" message from another node (i.e., its ID is included in this "I-HEAR-YOU" message), it identifies the latter as an outbound neighbor. The two nodes will become both inbound and outbound neighbors for each other, if they can hear both "HELLO" and "I-HEAR-YOU" messages from each other (i.e., the link is symmetric).

As link quality varies, a node uses the ratio of "HELLO" packets it received from each of its neighbors as an estimation of inbound link quality (inbound PRR). This inbound PRR value is then also put in the "I-HEAR-YOU" packet (together with the inbound neighbor ID) and broadcast back to the corresponding neighbor, so that the neighbor is able to be aware of its outbound link quality (outbound PRR) to this node.

Note that in the widely adopted implementation of BAP, the "I-HEAR-YOU" message, which contains the list of inbound neighbors as well as their PRR values, and the "HELLO" message, are often combined into one packet. We refer to these dual-role broadcast packets as *beacons*.

B. Asymmetric Link Discovery and Exploitation using BAP

As introduced in previous section, BAP is designed for symmetric links. It has two main drawbacks when facing asymmetric links.

Firstly, BAP cannot effectively convey information over the poor links. Consider an example where the link between node A and node B is asymmetric (A to B is good direction). B can detect that A is its good inbound neighbor, as B can receive A's beacons. In addition, B can identify that A is not its outbound neighbor, as B does not find itself in A's beacons. As a result, B is aware of the existence of the asymmetric link.

Unfortunately, under the BAP scheme, it is hard for A to be aware of the existence of asymmetric links, because the beacons ("I-HEAR-YOU" messages) from B rarely reach A. For the same reason, it is also hard for A to maintain the instantaneous outbound PRR to B, although B is able to continuously estimate that its inbound PRR from A is good.

Secondly, in an ACK-enabled network, the link quality is not only defined by outbound PRR, but also the inbound ACK reception ratio (ARR). For example, the widely adopted link metric for link quality based routing, expected transmission count (ETX) [12] is defined as $\frac{1}{\text{PRR}_{A\rightarrow B}} \times \frac{1}{\text{ARR}_{B\rightarrow A}}$ (ETX from node A to node B).

However, BAP can only measure PRR. It cannot measure ARR directly. Existing approaches adopting BAP try to infer ARR from PRR, including (1) assuming ARR $=$ 100% as in TinyOS 1.x and [4]; (2) assuming ARR $=$ PRR as in TinyOS 2.x and [12]; and (3) using experimental model, such as ARR $=$ PRR$^{S_{\text{ACK}}/S_{\text{PKT}}}$ in [10]. When PRR is high (the symmetric link), these three approaches make no big differences. However, when PRR is small (the poor direction of an asymmetric link), the estimation errors can be large. Although Figure 2 shows that the third estimation can be more accurate than the first two, it may not apply to all wireless environments. Especially, when PRR is low, ARR has large variation, and cannot be inferred accurately using PRR.

C. Information Feedback via Multihop Relaying

In order to convey the information from B to A (the poor direction of an asymmetric link), the most natural way is to let one (or many) intermediate nodes between A and B participate in the relay of this information. Previous works employed mechanisms including controlled flooding, unicast routing and volunteer relaying, which will be described below.

1) Controlled Flooding: Controlled flooding [6] is probably the simplest way to feed back information from B to A. When B detects the existence of the asymmetric link, it simply floods the "I-HEAR-YOU" packet to its k-hop neighbors, by setting TTL (Time-To-Live) of the packet to k. If a path exists from B to A within k hops, the "I-HEAR-YOU" packet can reach A through the path. However, the multi-hop broadcast is essentially "blind" and notorious for wasting both energy and

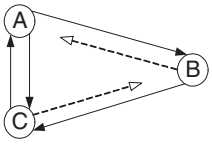


Fig. 3. Asymmetric link discovery

network capacity. Making things worse, in order to maintain the time-varying link quality or update A about B's current routing metric, the controlled flooding needs to be carried out regularly.

2) Unicast Routing: Most stateful n-to-n routing protocols such as AODV can provide shortest paths from a node to any other nodes in the network. Utilizing network layer information, it is possible for B to unicast the asymmetric link discovery and maintenance information to A along the shortest path from B to A [6]. This scheme is both bandwidth and energy efficient compared to controlled flooding. However, stateful routing protocols require each wireless node to maintain states for all other nodes in the network, which is often too expensive to be implemented in low power sensor networks.

This approach can also lead to a deadlock situation where some routes exist only when some asymmetric links are discovered, while the discovery of those asymmetric links requires the routes to exist first.

Consider the case of three nodes A, B and C shown in Figure 3. The link between node A and C is symmetric link. The link between A and B is asymmetric (A to B is good), and the link between B and C is asymmetric (B to C is good). When B finds that A is a good incoming neighbor, and wants to forward this information to A, it can not find any route to A, unless the link from B to C is discovered first. However, when C wants to tell B the information about the link from B to C, it can not find any route to B, unless the link from A to B is discovered first. This deadlock will result in neither of the two asymmetric links being discovered. We will show in the next section that this can be solved by our proposed Source Specified Relaying (SSR) scheme, because it uses only link layer information.

3) Volunteer Relaying: In [4], Sang et al. propose volunteer relaying for information feedback. In their scheme, each node need to monitor all links of its 1-hop neighbors to identify asymmetric links among them. Again take Figure 3 as an example. Node C can be aware of the fact that the link between A and B is asymmetric, as C can receive both A and B's beacon. By examining their beacons, C finds that B's beacon includes A while A's beacon does not include B. If C also knows that its link quality to A is good, it will volunteer itself to relay the link discovery and maintenance information to A for B.

This scheme is inefficient in two ways. First, suppose each node has n neighbors, this requires the node to keep states for $O(n^2)$ links, which can be expensive in a dense sensor network. Second, due to the fact that more than one neighbors may volunteer themselves, the scheme can cause unnecessary duplicates. Though suppression techniques may be employed to reduce duplication, the performance will rely on the efficiency of the suppression technique, and all volunteer nodes need to keep even more states for suppression.

Also, implementation wise, this scheme is sensitive to the predetermined parameter of relaying threshold. C volunteers itself only when its link quality (PRR) to A is good enough. The given implementation sets the threshold value to 80%. If this threshold is set too low, a non-optimal neighbor (with low PRR to A) may volunteer itself to help A before a better node volunteers. If the value is set too high, even a good enough neighbor may be disqualified and the link will not be discovered.

D. Unicast Probing for ARR Measurement

One way to resolve the inaccurate ARR estimation in BAP is to use unicast probing. Unicast based link quality measurement is the most accurate scheme because it measures the link quality (PRR $\times$ ARR) in the exact way that the link is being used. However, unicast probing incurs per-pair overhead in link estimation, which does not scale as well as BAP. The overhead of unicast probing may be reduced by techniques such as opportunistically exploiting real traffic and cross traffic, as suggested in [3].

IV. Asymmetry-Aware Link Discovery and Exploitation

As stated in the previous section, although BAP is efficient in discovering and exploiting symmetric links, it has various drawbacks when dealing with asymmetric links. In this section, we propose an efficient link discovery and exploitation scheme. In our work, BAP is still used for symmetric link discovery and exploitation. We propose two techniques to work together with BAP and efficiently resolve the deficiencies of BAP.

First, we introduce an efficient information feedback scheme over poor links called *Source Specified Relay (SSR)*. SSR opportunistically makes use of the local information at the link layer for asymmetric link discovery and exploitation. Unlike the existing relaying protocols which have either large communication cost (controlled flooding), or large memory cost (unicast routing and volunteer relaying), SSR needs almost no extra communication and memory cost. Second, we propose *Dynamic Driven Maintenance (DDM)* which adaptively switches between BAP, SSR-enabled BAP and unicast probing for link maintenance under different link conditions. ARR can also be more accurately estimated by adopting different schemes under different conditions.

In the rest of this section, we will present SSR and DDM in detail.

A. Source Specified Relay (SSR)

SSR serves mainly as an information feedback mechanism over the poor direction of an asymmetric links. It opportunistically makes use of the local information at link layer to find the

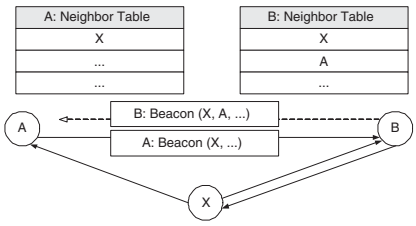


Fig. 4. Source specified relaying

relay nodes for information feedback over the poor direction of the asymmetric link.

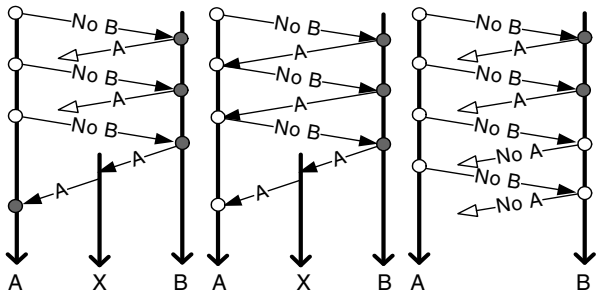
SSR is better explained using the example shown in Figure 4. Let the link between A and B be asymmetric (A to B is the good direction). B needs to find a common neighbor (X) of A and B who can help relay the neighbor discovery message from B to A. If possible, X shall be the best among all common neighbors of A and B considering both the link qualities from B to X and X to A. In other words, X must be in both A and B's neighbor tables and has the largest product of PRR from B to X (B's outbound link quality) and PRR from X to A (A's inbound link quality).

As shown in Figure 4, B knows who are its neighbors as well as its outbound link quality to these neighbors. The outbound link quality of its neighbors are obtained via link quality feedback from these neighbors through BAP.

On the other hand, B has to know A's neighbor as well as the inbound link quality from A's neighbor to A before it can decide who is the best common neighbor for relaying. Fortunately, this information is provided by A's link quality feedback beacons to A's neighbors. Because the link quality from A to B is good, B can consistently hear A's link quality feedback beacons. It is then possible for B to check the intersections (common neighbors) of its own neighbor table and A's feedback beacon packets and find the best X. After X is identified, B can directly request X to relay its beacon to A.

In SSR, B does not need to maintain A's neighbor table because the table is broadcasted by A periodically in A's beacons. Upon inserting A into its table, B can already begin to search for X every time B receives a beacon from A. Thus, once SSR is triggered, B can immediately use the identified optimal relay node to forward packet to A. This *identify-while-wait* scheme is especially useful when it takes the other node (A) several beacons to broadcast its entire neighbor table. Note that, as all of the above information are already available in BAP, there is almost no extra cost. This process is illustrated in Figure 5(a). Note that SSR does not require the link between A and X to be symmetric links. Thus, a just discovered asymmetric link can be used to discover new asymmetric links.

Consider the example in Figure 3 again, when node C wants to feed back B about the asymmetric link between them, it will intersect B's inbound neighbors with its outbound neighbors, and find A as the relay node. Once the asymmetric link between B and C is discovered, B can use C to forward beacon to A. Thus, the asymmetric link between A and B can also be discovered.



(a) Normal Discovery (b) False Discovery (c) Early Eviction

Fig. 5. Triggering of asymmetric link discovery. Solid arrow: the packet is received successfully. White arrow: the packet is lost. Shaded circle: the other side is in neighbor table. White circle: the other side is not in neighbor table.

The weakness of SSR is similar to the volunteer relaying scheme. It works only if X exists. Our evaluation in Section VI shows that, the existence probability of relay node is high (more than 80%) as long as the network density is not too sparse. Note that there are cases where the asymmetric links cannot be discovered by SSR (and volunteer routing) even though they can be discovered by two-hop flooding. For example, if the link between A and C in Figure 3 is also an asymmetric link (C to A is good direction), both SSR and volunteer routing cannot discover any of the three asymmetric links, while two-hop flooding can.

B. Dynamic Driven Maintenance (DDM)

Based on discussions in previous sections, it can be seen that the three schemes (BAP, BAP with relaying feedback, Unicast probing) are suitable for different scenarios. In particular, BAP should be used when the links are good in both directions, relaying feedback (SSR) should be used together with BAP when BAP alone is not efficient for link quality feedback over poor links, and unicast probes should be applied when PRR is so small that ARR cannot be accurately estimated.

DDM tradeoffs accuracy and efficiency by dynamically and adaptively using broadcast probing (BAP), SSR-enabled BAP and unicast probing.

The design of DDM is motivated by the fact that the asymmetric link can be a temporal phenomenon. The poor direction's PRR can vary. When PRR of an asymmetric link over the poor direction (from B to A in the example above) is higher than some threshold (B is able to know this information from A's beacons), say Q_{good} (default to 50% in our setting, below which direct feedback is less efficient than relaying), the link is more likely to be a symmetric link, BAP is sufficient for link maintenance. The ARR can also be inferred with relatively high accuracy from PRR.

On the other hand, if the PRR (from B to A) is lower than the threshold Q_{good}, DDM chooses to maintain the link using broadcast but with the help of SSR to improve information feedback efficiency.

Lastly, in an ACK-enabled network, when PRR from B to A drops below a threshold, say Q_{bad} (default to 20%, below which the ARR cannot be accurately estimated directly from PRR), node A transits to use unicast probing to maintain the

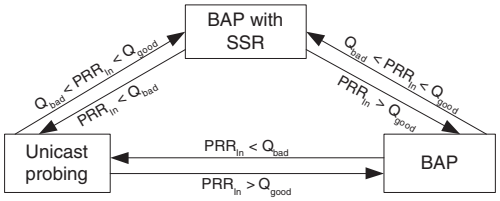


Fig. 6. DDM flow with respect to a single neighbor

forward link to node B. In this situation, the ARR over the poor direction (from B to A) can be measured much more accurately.

Note that, when unicast probing is used for link quality measurement (the PRR $\times$ ARR from A to B), B still needs to update A about its own routing metric. SSR can be used for the routing metric update. More details about this will be discussed in Section V-C.

The control flow of DDM with respect to a single neighbor is shown in Figure 6. To improve stability, our implementation includes a hysteresis threshold in triggering the state change.

V. Memory Constraints and Asymmetry-Aware Caching (AAC)

Typical WSN platforms have limited memory. Most sensor node designs have just a few kilobytes of RAM. For example, the MICA2 and MICAz motes have only 4KB of RAM, and the Telosb motes have only 10KB of RAM. This, in turn, imposes limits on the storage requirements of network (and other) protocols, requiring routing tables, buffering, and caches be kept small. The historical trends of monetary and energy costs suggest these constraints are likely to last [13].

As a result, the amount of resource dedicated to neighborhood management should also be very small. As such, the default neighbor table sizes are 16 and 10 in TinyOS 1.x and 2.x respectively. Such limitations often mean the motes cannot store state for all possible neighbors [14]. In addition, in order to reduce the complexity, the link status (and routing) information must be maintained in a table with constant space regardless of cell density [11]. The use of small table size also helps to reduce state maintenance/computation and communication overhead needed to keep the state information up-to-date.

Therefore, even with good asymmetric link discovery and proper link maintenance, it is essential that the neighbor table management is also aware of asymmetric links, in particular, when using a small, fixed size neighbor table. Improper neighbor management can cause symmetric links to be incorrectly identified as asymmetric links (false discovery), and asymmetric links to be evicted before it is discovered (early eviction). In this section, we propose *Asymmetry-Aware Caching (AAC)* to work with SSR and DDM proposed in previous section, so that both symmetric and asymmetric links are discovered and exploited efficiently. The name AAC comes from the fact that the neighbor table functions like a kind of cache of desirable links [11].

A. False Discovery

As mentioned in Section III, for an asymmetric link between node A and B (A to B is good direction), B triggers asymmetric link discovery (SSR) when it finds itself not in A's beacons. However, if neighbor table size is limited, the fact that B is not in A's feedback beacons does not necessarily imply A cannot hear B. It can also be because A has no space to maintain B in its neighbor table. We call the phenomenon that the link quality from B to A is sufficiently good while B still triggers the asymmetric link discovery as a *false discovery*. This is shown in Figure 5(b). False discovery is not desirable because it introduces unnecessary forwarding overheads.

AAC suppresses false discovery by adding a small field in each node's beacon: *table entry threshold*. This threshold is computed as the smallest (outbound) PRR of all neighbors kept in the table. For a given node A, if (and only if) a new neighbor B's quality (in terms of A's PRR to B) is higher than A's *table entry threshold*, B is guaranteed to enter A's table (possibly with some existing entry in the table evicted).

Under AAC, before a node B sends feedback to a newly discovered neighbor A, it first compares the PRR from A with the latest table entry threshold announced by A. If the PRR from A is lower than the threshold, B suppresses the sending of feedback. This is because, even if the feedback is sent, it will not be used by A. In contrast, if the PRR from A is higher than A's table entry threshold, B puts A (with PRR) in feedback beacons. When A receives this feedback, it puts B into table, and feeds back B. Thus, if B does not find itself in A's feedback beacon after it sends out its feedback to A, it can safely infer that its feedback is not received by A. When such situation happens repeatedly, B starts the asymmetric link discovery using SSR.

Broadcasting table entry threshold can greatly reduce the probability of false discovery under neighbor table size constraints. In addition, note that even if a symmetric link is falsely identified as asymmetric link, DDM can ensure that it quickly switches back to BAP maintained states.

B. Early Eviction

If the reverse link is good, broadcast-based discovery is less expensive than SSR discovery as the latter incurs unnecessary relaying. Thus, when one node begins to feedback a new neighbor, it will firstly use broadcast feedback. SSR is triggered only when the neighbor fails to be notified after several retrials. During this testing period, the outbound link quality towards that new neighbor is unknown. When neighbor table size is limited, such neighbor is easily evicted from table by existing neighbor table schemes. If an asymmetric link is evicted during testing period (before SSR is invoked), we call it an *early eviction*.

For example, in TinyOS 1.x implementation, when the neighbor table overflows, the neighbor with least outbound PRR is evicted. In TinyOS 2.x implementation, the neighbor with least product of inbound and outbound PRR is evicted. In both cases, a neighbor without outbound link quality is easily evicted, even if its inbound link quality is high. Early eviction

reduces the number of asymmetric links found, and increases the delay for asymmetric link discovery.

AAC reduces early eviction by preventing a neighbor from being removed before its status can be clearly determined. Due to lack of space, the detail logic and state transition will not be presented here.

C. Stateless Routing Metric Feedback

A final problem with neighbor table size constraint is that when SSR is to be applied on routing metric feedback during maintenance stage as shown in Figure 4, the receiving end of the good link, in this case node B, must keep node A in its neighbor table. Otherwise, when B's routing metric changes, node B does not know that it needs to update A. This is because that the broadcast-based method for routing metric update does not work well due to the asymmetric link, and B has to use SSR to send its routing metric to A.

Note that even if B maintains A's state, which is not desirable when the neighbor table size is limited, it is still not able to execute SSR immediately, because it needs time to find the relay node X to node A. Depending on the length of interval that node A takes to dump its whole neighbor table into feedback beacons, the delay can be quite large.

When to trigger B to send feedback, and how to find the relay from B to A are two issues address by the *Stateless Routing Metric Feedback* scheme. Here stateless refers to the fact that B does not need to maintain any state about A.

In the scheme, we take advantage of the fact that, when B evicts A out of its neighbor table, node A must use unicast probing in DDM to maintain the link quality. We achieve stateless routing update by opportunistically using the (free) unicast probing packets from A to B. Node A puts two additional information into the unicast packet: the latest link/routing information of B as known by A, and the identity of node X that can serve as a relay from node B to node A. Upon receiving such a unicast probe from A, node B checks whether the information as reported by A is outdated, and will start a relayed feedback to A via X if it is outdated.

The remaining question is how A computes X. In fact, it is done in a similar way as SSR. The idea is shown in Figure 7. A is able to know that an inbound neighbor X is suitable for relaying, because A knows the inbound link quality from X to A (recorded in A's own neighbor table), as well as the link quality from B to X (announced in X's feedback beacons). To find the optimal relay X, A simply searches for B in each beacon that it receives from its neighbors.

VI. Performance Evaluation

We first evaluate the performance of SSR, followed by evaluation of AAC with constrained table size using simulation. Then, we study the performance of DEAL (SSR, DDM and AAC as a whole) using routing performance as the benchmark by both simulation and testbed evaluation.

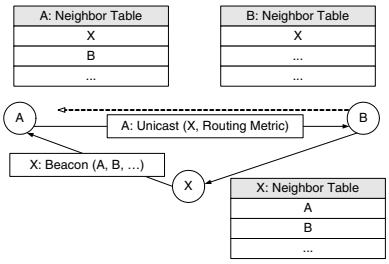
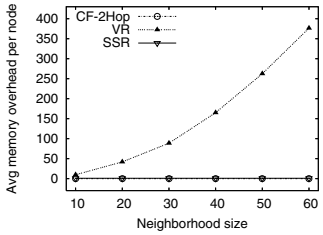
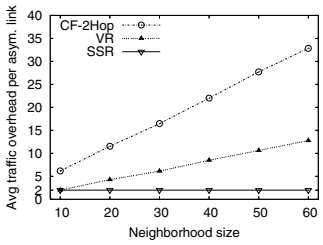


Fig. 7. Stateless routing metric feedback



(a) Memory overhead



(b) Communication overhead

Fig. 8. Performance of SSR

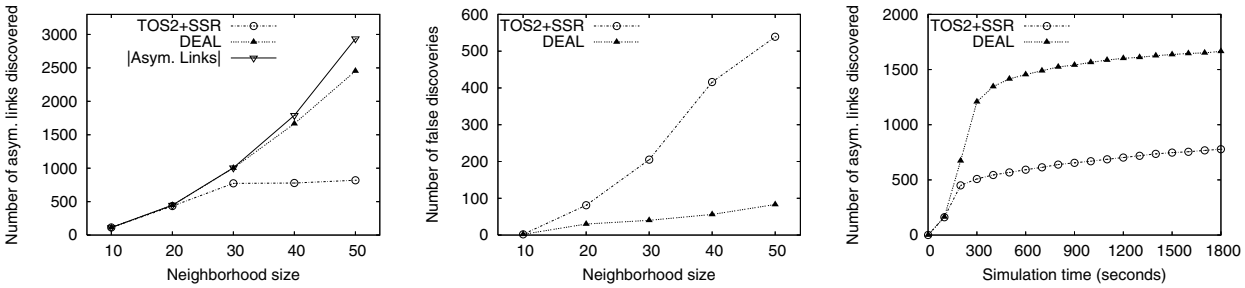
A. Evaluation of SSR

We evaluate the performance of SSR using TOSSIM [15]. In the simulation, nodes are generated randomly in a $150m \times 150m$ square field and link qualities among nodes are generated based on the empirical measurements collected from our MICA2 testbed. More specifically, we group the links from our testbed by their distance with a granularity of $5m$. The qualities of simulated links are then sampled from the corresponding bins. We assume there is no link between two nodes if they are more than $60m$ apart.

The performance of SSR is compared with 2-hop controlled flooding (CF-2Hop) and volunteer relaying (VR) using three metrics: percentage of asymmetric links that can be discovered, memory required at each node and communication overhead. The neighborhood size is varied by changing the number of nodes generated. Each point is the average of 100 runs.

In terms of percentage of asymmetric links found, the result shows that 2-hop controlled flooding discovered 99.5% of the links, SSR 94.5% and VR 82%. SSR discovers more asymmetric links than VR, because when there is no relay nodes with PRR higher than 0.8, SSR automatically finds the optimal relay node among those available, while in VR, no node will volunteer as relay.

Figure 8(a) plots the average memory required per node over different neighborhood sizes. Controlled flooding does not



(a) Asym. link found vs. neighborhood size (b) False discoveries vs. neighborhood size (c) Asymmetric link discovery time

Fig. 9. Performance of AAC

require any additional memory consumption. As SSR needs to record selected relay nodes in the neighbor table, it consumes a small amount of memory for each table entry. On the other hand, in order to volunteer their services, nodes in VR have to monitor all links in their neighborhood. As a result, it incurs the highest memory overhead. In fact, when neighborhood size increases by n, the memory overhead of VR can increase by $O(n^2)$.

Figure 8(b) plots the average communication overhead incurred to discover one asymmetric link. The communication overhead for 2-hop controlled flooding is the highest since all neighboring nodes forward the packet. For volunteer relaying, Figure 8(b) shows the worst case average communication overhead per asymmetric link, as we do not employ any relay suppression. The value shown is the number of duplicate relay nodes. For SSR, as only one relay node is selected, the communication overhead per link is constant.

Overall, the results show that SSR is able to discover most of the asymmetric links with very low memory and communication overhead compared to CF-2Hop and VR.

B. Evaluation of AAC

This section evaluates the performance of DEAL when the neighbor table size is constrained. Simulation setup is the same as the previous section. To make a fair comparison, we implemented SSR in TinyOS 2.x and compare it to DEAL (SSR+DDM+AAC). Therefore, we allow TinyOS 2.x to make use of SSR for asymmetric link discovery, while using its default neighbor table management scheme. In the simulation, the neighbor table size is set to 20. (This is larger than the default values of 16 in TinyOS 1.x and 10 in TinyOS 2.x). The average neighborhood size is varied from 10 to 50.

Figure 9(a) shows the number of asymmetric links discovered by DEAL and TinyOS 2.x with SSR. When the neighborhood size is smaller than 20, the two protocols have similar performance, where most of the asymmetric links are discovered. This is because the neighbor table size is larger than the average neighborhood size and there is very little overflow in the neighbor table.

However, when the neighborhood size increases further, the performance of TinyOS 2.x with SSR drops rapidly. In fact, the total number of asymmetric links discovered does not increase when the average neighborhood size increases beyond 30. This is because most asymmetric links are evicted from the table before they can be discovered.

On the other hand, with a better table management policy, DEAL can continue to discover more asymmetric links even as the average neighborhood size increases to 50, more than 2 times the neighbor table size of 20.

Figure 9(b) shows the number of false discoveries (as defined in Section V-A) for TinyOS 2.x with SSR as well as DEAL. The results clearly show that DEAL has a much smaller number of false positives compare to TinyOS 2.x. Taking the results of the figures together, when the average neighborhood size is large, say 50, DEAL discovers over 2500 asymmetric links with only 100 false positives, while TinyOS 2.x with SSR discovers about only 800 asymmetric links with about 550 false positives!

Figure 9(c) shows the number of asymmetric link discovered over time, when the neighborhood size is 40. It can be seen that DEAL discovers more asymmetric links in a shorter time. After 300 seconds (30 beacon intervals), DEAL discovers more than 80% of asymmetric links, compare to around only 30% for TinyOS 2.x.

C. Routing Performance Evaluation of DEAL

In this section, we evaluate the end-to-end performance of DEAL using a data collection tree application and ETX (products of PRR and ARR) as the routing metric. For performance comparison, we use the default TinyOS 2.0 implementation using the product of PRRs as routing metric (TOS2+PRR), and the default TinyOS 2.0 implementation using ETX as routing metric (TOS2+ARR). Note that the same default TinyOS routing algorithm is used in all the three schemes.

Figure 10 plots the average path ETX values for the three schemes with increasing average neighborhood size. The result shows that for different neighborhood sizes, DEAL outperforms the other two schemes. The average reduction in path ETX is 25% and 10% over TOS2+PRR and TOS2+ARR respectively. The performance gap between TOS2+PRR and TOS2+ARR shows the importance of estimating ARR accurately (the purpose of incorporating DDM into DEAL). Similar performance improvement can also be observed for end-to-end data transmission delay and average hop count.

Figure 11 plots the ETX reduction comparing DEAL and TOS2+ARR of different runs for the case when the average neighborhood size is 20. It can be seen that the average path

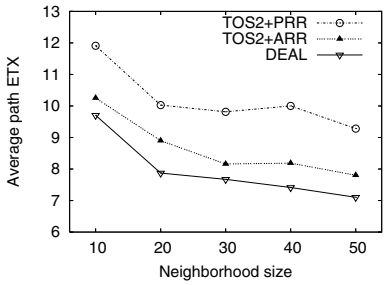


Fig. 10. ETX reduction vs. neighborhood size

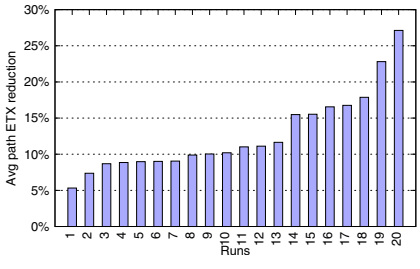


Fig. 11. ETX reduction for each run

ETX can be reduced by up to 20% in some cases. In cases with substantial ETX reduction, there exist a number of long asymmetric links pointing towards the sink. On the other hand, in cases where the improvement is small, the asymmetric links tend to be pointing away from the sink, making them less useful.

D. Testbed Evaluation

In this section, we compare the performance of the default implementation of TinyOS 2.x and DEAL on a Telosb (which has the same radio as MICAz) testbed with 16 nodes. The testbed is installed in an indoor area spanning over a space of $1000m^2$, and each node is connected to the AC power. The nodes are mounted from the ceiling and there is line-of-sight between most pair of nodes. The nodes can be remotely accessed for easy experimental data collection.

When sensor nodes are battery-powered, the batteries on these nodes drain at different rates. Typically, nodes closer to the sink encounter faster energy depletion rate. Our measurements on battery powered sensor nodes indicated that as the battery power drained, PRR decreases. In order to simulate such effect, we vary the transmission power on selected nodes.

Starting from the situation where all nodes are using high power ($-15dBm$), we randomly pick one additional node to reduce its power to $-25dBm$. Nodes closer to the sink tends to be picked earlier. With 16 nodes, there are 17 sets of experiments. Each experiment is run for 2 hours.

The result is shown in Figure 12. It can be seen that DEAL generally achieves better performance (in terms of lower ETX) than the default TinyOS 2.x implementation. This is especially true when the nodes around the sinks start using low power. The largest improvement in path ETX is about 21% comparing to TinyOS 2.x. The better performance of DEAL can be attributed to the fact that adoption of asymmetric links decreases the average hop count of the network.

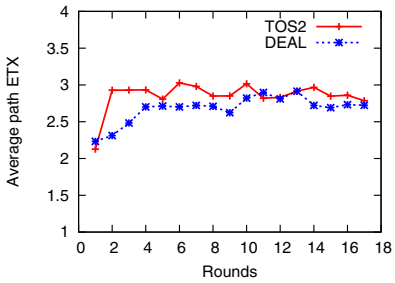


Fig. 12. Testbed evaluation

VII. Conclusion

This paper systematically addresses the problem of exploiting asymmetric links in link layer. We propose DEAL to discover and maintain asymmetric links both accurately and efficiently. DEAL consists of SSR, a scheme for information feedback over the poor direction of an asymmetric link; DDM, a scheme for both efficient link maintenance and accurate ARR estimation; and AAC, a scheme for limited neighbor table size handling. We implement DEAL in TinyOS, and study its impacts on a tree collection application. Simulation and evaluation results show that more than 80% of asymmetric links are discovered and exploited with minimum overhead, which improves routing layer packet delivery efficiency by up to 20%. Testbed evaluation also shows that DEAL improves the network routing performance by identifying useful asymmetric links.

References

[1] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, and D. Estrin, "Complex behavior at scale: An experimental study of low-power wireless sensor networks," UCLA/CSD, Tech. Rep., 2002.
[2] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *ACM Sensys03*.
[3] K.-H. Kim and K. G. Shin, "On accurate measurement of link quality in multi-hop wireless mesh networks," in *ACM Mobicom'06*.
[4] L. Sang, A. Arora, and H. Zhang, "On exploiting asymmetric wireless links via one-way estimation," in *ACM Mobihoc'07*.
[5] G. Zhou, T. He, S. Krishnamurthy, and J. Stankovic, "Impact of radio irregularity on wireless sensor networks," in *ACM Mobisys'04*.
[6] M. R. Pearlman, Z. J. Haas, and B. P. Manvell, "Using multi-hop acknowledgements to discover and reliably communicate over unidirectional links in ad hoc networks," in *WCNC'00*.
[7] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11b mesh network," in *ACM Sigcomm'04*.
[8] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, "Understanding the causes of packet delivery success and failure in dense wireless sensor networks," in *ACM Sensys'06*.
[9] K. Srinivasan and P. Levis, "Rssi is under appreciated," in *EmNets'06*.
[10] S. Lee, B. Bhattacharjee, and S. Banerjee, "Efficient geographic routing in multihop wireless networks," in *ACM MobiHoc'05*.
[11] A. Woo, T. Tong, and D. Culler, "Taming the underlying challenges of reliable multihop routing in sensor networks," in *ACM Sensys'03*.
[12] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *ACM Mobicom'03*.
[13] P. Levis, E. Brewer, D. Culler, D. Gay, S. Madden, N. Patel, J. Polastre, S. Shenker, R. Szewczyk, and A. Woo, "The emergence of a networking primitive in wireless sensor networks," *Communication of the ACM*, vol. 51, no. 7, pp. 99–106, July 2008.
[14] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis, "Four-bit wireless link estimation," in *HotNets'07*.
[15] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: accurate and scalable simulation of entire tinyOS applications," in *ACM Sensys'03*.