Timely Data Delivery in Sensor Networks using Whirlpool

Divyasheel Sharma
Info Sc & Tele Dept
University of Pittsburgh
PA 15260 USA
1-412-624-7378
dsharma@sis.pitt.edu

Vladimir I. Zadorozhny
Info Sc & Tele Dept
University of Pittsburgh
PA 15260 USA
1-412-624-9411
vladimir@sis.pitt.edu

Panos K. Chrysanthis
Computer Sc Dept
University of Pittsburgh
PA 15260 USA
1-412-624-8925
panos@cs.pitt.edu

ABSTRACT
In this paper, we introduce a *rotating interrogation* technique for sensor networks called *Whirlpool. Whirlpool* provides opportunities to optimize data delivery in time-critical monitoring applications. We explore concurrency opportunities while delivering data using the whirlpool strategy and investigate several whirlpool algorithms. We also demonstrate extra benefits of using whirlpool with our algebraic optimization framework based on Data Transmission Algebra.

Categories and Subject Descriptors
C.2.1 [**Computer – Communication Networks**]: Network Architecture and Design – Wireless communications; H.2.4 [**Database Management**]: Systems – Distributed databases, Query processing.

General Terms
Algorithms, Design, Performance, Experimentation

Keywords
Sensor Networks, Collision Domains, Query Optimization, Data Transmission Algebra, Whirlpool

1. INTRODUCTION
Recent advances in wireless communications and microelectronics have enabled wide deployment of smart sensor networks. Such networks naturally apply to a broad range of applications that involve system monitoring and information tracking (e.g., airport security infrastructure, monitoring of children in metropolitan areas, product transition in warehouse networks, fine-grained weather/environmental measurements, etc.). Meanwhile, there are obvious performance deficiencies in applying existing sensornets (SNs) in *mission-critical monitoring applications*, such as Structural Health Monitoring (SHM) [1]. Because a system failure in such applications is often catastrophic, the associated costs are very high. This results in special requirements with respect to efficient mechanisms for querying sensor data and delivering the query result from a dense sensor network in a timely manner and without degradation in the quality of data. These requirements present distinctive challenge in the design of data delivery pattern in sensor networks. First, it calls for an efficient design that can maximize the *concurrent transmission* opportunities. Second, it should be able to deliver *sufficient amount of data* that can be used to deduce valuable information. Third, the design should have provisions for implementing energy efficient strategies. Fourth, the strategy should be scalable with increase in number of sensors. Finally, it should be robust to accommodate node failures.

DMSN'05, August 29, 2005, Trondheim, Norway.

In this paper we propose such a novel *whirlpool* data delivery technique. This technique tunes the sensor query processing for specific performance and quality of data requirements of mission-critical monitoring applications and keeps provisions for future development of the technique according to above mentioned characteristics. Whirlpool query processing is based on splitting the sensor network into *sectors* and performing a *rotating interrogation* over the sectors such that the complete network is monitored. This introduces a natural inter-sector parallelism when two or more sectors can be interrogated simultaneously without risk of *packet collisions*, - a major source of time and energy waste in wireless communication. The flexibility and controlled efficiency of whirlpool makes it an appropriate query processing technique for mission-critical monitoring application. We further combine whirlpool with an algebraic optimization framework that utilizes information about how the medium access control (MAC) layer operates while processing sensor queries [2]. In contrast with other approaches assuming that the MAC layer handles collisions in an appropriate manner, our optimizer performs collision-aware query scheduling that reduces the amount of signaling traffic and retransmissions. The core component of our framework is Data Transmission Algebra (DTA) [2, 3] that captures the structure of data transmissions along with their constraints and requirements.

Section 2 outlines some background and system model. Section 3 gives general description of the whirlpool strategy. Section 4 provides basic whirlpool algorithm and then extends it by exploring more concurrency opportunities in whirlpool. Section 5 provides experimental evaluation of the whirlpool algorithms. In Section 6 we consider related works. Section 7 concludes.

2. BACKGROUND AND SYSTEM MODEL
Consider a wireless sensor network deployed in order to monitor structural integrity. An example query over this network could request vibration data over a certain period of time. Answering

this query would result in a tree-like data delivery pattern (Figure 1). This implies that the transmissions between sensors are *ad hoc* dependent on the query and require the use of a medium access control (MAC) layer to handle transmissions on the same medium and a routing algorithm that enables the nodes to select the right neighbor to transmit data.

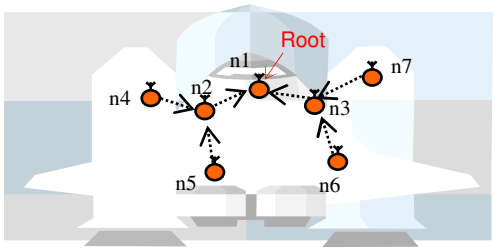

Figure 1. An example of a query tree

Popular wireless MAC layer technologies are the IEEE 802.11 standard for wireless local area networks [4] and the IEEE 802.15 standards for wireless personal area networks [5]. For low power and low data rate sensor networks, the 802.15.4 standard appears to be suitable. One issue, which is common for all MAC layer protocols, is proper handling of packet collisions. If we assume that all sensor nodes use the same frequency band for transmission, two transmissions that overlap will get corrupted (collide) if the sensor nodes involved in transmission or reception are in the same *collision domain* defined as the union of the transmission ranges of *the communicating nodes*. This is independent of the MAC protocol selected.

3. WHIRLPOOL DATA DELIVERY TECHNIQUE

In this section, we focus on describing *concurrency intensive* whirlpool technique that minimizes number of collisions in sensor monitoring systems. The basic idea of whirlpool consists in splitting the sensor network into *sectors*, as illustrated in Figure 2. The number and size of sectors can vary depending on the monitoring requirements. We define two sectors S1 and S2 as *colliding,* if at least one transmission of S1 collides with any transmission of S2. As assumed the adjacent sectors are always colliding. For example, in Figure 2b (S1, S2), (S1, S4), (S2, S3), and (S3, S4) are pairs of colliding sectors. Transmissions within a group of non-colliding sectors can be conducted concurrently except at the last hop as described below. This introduces natural *inter-sector concurrency*. In Figure 2b, (S1, S3) and (S2, S4) sector groups could be executed concurrently. Although some sectors can be scheduled concurrently, there is a one hop "serial bottleneck" near the whirlpool center, since all final elementary transmissions of each sector share the same destination (base station). Whirlpool algorithms that deal with this bottleneck are explained in the next section.

Whirlpool performs *rotating interrogation* in the sensor network. One whirlpool rotation is said to be *complete* if the base station receives a query response from each of the sectors. Multiple rotations constitute the case where responses from all the sectors are obtained two or more times.

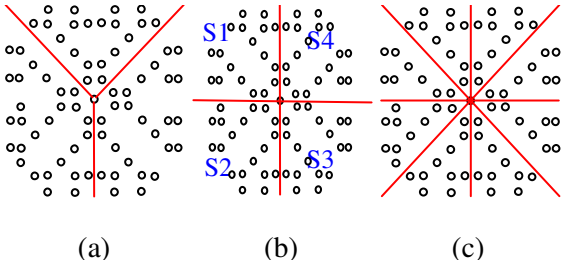

Figure 2. Sectoring of sensor network

3.1 Intra-sector Concurrency

In addition to intra-sector concurrency, whirlpool can also utilize an *intra-sector concurrency*. We achieve this by combining whirlpool with our algebraic optimization framework that utilizes information about how the medium access control (MAC) layer operates while processing sensor queries [2]. The core component of the framework is Data Transmission Algebra (DTA) [2, 3]. The DTA consists of a set of operations that take transmissions between wireless sensor nodes as input and produce a schedule of transmissions as their result. We call an *elementary transmission* (denoted *ni~nj*) a one-hop transmission from sensor node *ni* to node *nj*. Each transmission *ni~nj* is associated with a collision domain CD(*ni*, *nj*) as defined in Section 2. A transmission schedule is either an elementary transmission or a composition of elementary transmissions using one of the operations of the DTA. The basic DTA includes three operations that combine two transmission schedules A and B:

- ***o*(A,B)**. This is a strict order operation, that is, A must be executed before B.
- ***a*(A,B)**. This is an overlap operation, that is, A and B can be executed concurrently.
- ***c*(A,B)**. This is a non-strict order operation, that is, either A executes before B, or vice versa.

DTA can be used to enhance our whirlpool interrogation strategy with intra-sector concurrency. For example, consider a query tree in Figure 3 that corresponds to one sector of a whirlpool. The figure also shows the *initial DTA specification* reflecting basic constraints of the query tree. For instance, operation *o(n4~n2, n2~n1)* specifies that transmission *n2~n1* occurs after *n4~n2* is completed because of the query tree topology. Operation *c(n2~n1, n3~n1)* specifies that there is an order between transmissions *n2~n1* and *n3~n1* since they share the same destination. However this order is not strict. Operation *a(n4~n2, n5~n3)* specifies that *n4~n2* can be executed concurrently with *n5~n3*, since neither *n3* nor *n5* belongs to CD(*n4,n2*), and neither *n4* nor *n2* are in CD(*n5,n3*).

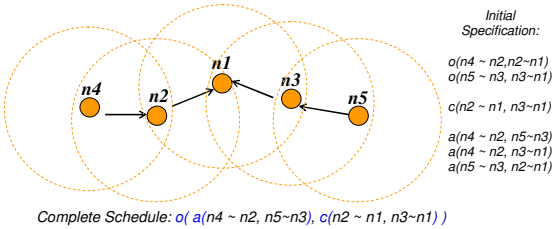

Figure 3. Example of DTA specifications

The DTA also includes a set of transformation rules [2] that can be used to generate complex transmission schedules. Figure 3 shows an example of a complete schedule that involves all elementary transmissions of the sector. Our optimizer performs cost based selection of the best schedule that maximizes the intra-sector concurrency [3].

3.2 Whirlpool Tuning

The whirlpool can be fine-tuned with respect to the *number and size of sectors*, as well as the *number and speed of rotations*.

Number and size of sectors: As the number of sectors increases and the number of nodes in each sector decreases, query scheduling becomes more efficient. In case of using DTA, query scheduling becomes less complex as the optimizer applies DTA to smaller sectors. This results in a higher degree of intra-sector concurrency. Meanwhile, the larger number of sectors increases chances of *inter-sector concurrency*. However, while considering DTA based approach smaller sectors can also reduce the number of potential "good" query schedules explored by DTA. Whirlpool should be tuned for an optimal number and size of sectors so as to utilize the performance benefits of specific scheduling framework while keeping the scheduling complexity reasonably low.

Number and speed of rotations: Higher numbers of rotations imply that more data is collected from each sector. The number of rotations should be tuned on the basis of application requirements. The most important parameter of whirlpool is its *speed of rotation*. Speed of rotation can be considered as the time spent by whirlpool in each sector for one rotation. It can also be interpreted as the size of data sample received from each sector during one rotation. In general, the speed of rotation is higher for whirlpools with smaller sectors. Faster rotation would also assume lower quality of data. By tuning the whirlpool rotation speed, we can either deliver approximate data faster or accurate data slower.

To summarize, whirlpool introduces the following major advantages:

- *Complexity Reduction in Scheduling*
- *Introduction of Inter-sector Concurrency*
- *Increase of Network Utility in terms of Control over the Behavior*

The last item requires more explanation. We define sensor network utility as the ability to provide a better quality of data relevant to understanding the behavior of the monitored system within shorter periods of time. Variable Quality of Data (QoD) that achieved by whirlpool provides additional opportunities in trading query response time, energy and QoD, which is especially important in mission-critical applications such as structural monitoring for early instability detection. Using the whirlpool monitoring system we can localize the damage in particular network sectors and also evaluate propagation of unstable behavior.

4. WHIRLPOOL ALGORITHMS

4.1 Basic Algorithm: Serial Last Hops

Consider a whirlpool structure with 4 sectors S1, S2, S3, S4 and a base station placed in the center of the network (Figure 4). We assume that any whirlpool sector has a single *last hop transmission* that reaches the base station. In Figure 4, t_{1lh} is a last hop transmission of S1, t_{2lh} is the last hop transmission of sector S2 and so on.

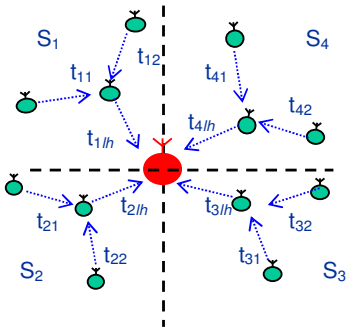

Figure 4. Whirlpool with 4 sectors

A *complete* sector schedule is a DTA schedule that includes all elementary transmissions of the sector[1]. For example, $o(c(t_{11}, t_{12}), t_{1lh})$ is a complete schedule for sector S1. A *sub-complete* sector schedule is a DTA schedule that includes all elementary transmissions of the sector except its last hop transmission. For example, $c(t_{11}, t_{12})$ is a sub-complete schedule for S1. Figure 5 represent the basic whirlpool algorithm (BA). First, the non-conflicting groups are formed. For the example in Figure 4 *NCGrp* = [{S1,S3}, {S2,S4}]. Then the schedules are grouped by non-conflicting sectors as follows:

$$Sch = [\{c(t_{11}, t_{12}), c(t_{31}, t_{32})\}, \{c(t_{21}, t_{22}), c(t_{41}, t_{42})\}]$$

Each of the sector groups *Sch[1]* and *Sch[2]* are executed in the order of whirlpool rotation. Meanwhile sub-schedules within each group are executed concurrently. We can express this inter-sector concurrency using DTA overlap operation *a* as shown in the following expression:

$$o(\, a(\, c(t_{11}, t_{12}), c(t_{31}, t_{32})\,),\ a(\, c(t_{21}, t_{22}), c(t_{41}, t_{42})\,)\,) \quad \textbf{(1)}$$

The above DTA expression schedules all sub-complete schedules of the whirlpool in Figure 4. In order to complete the scheduling we have to take care of last hop transmissions. Since all last hop transmissions share the same destination, the optimizer schedules them serially using DTA non-strict order operations:

$$LHSch = c(t_{1lh}, c(t_{2lh}, c(t_{3lh}, t_{4lh}))) \quad \textbf{(2)}$$

Combining schedules (1) and (2) using strict order operation we obtain a DTA expression for one whirlpool rotation:

[1] We can use any efficient scheduling technique here. Whirlpool is orthogonal to DTA in this sense.

o (o(a(c(t_{11} , t_{12}),c(t_{31} , t_{32})),

a(c(t_{21} , t_{22}),c(t_{41} , t_{42}))) ,

c(t_{1lh} ,c(t_{2lh} ,c(t_{3lh} , t_{4lh})))).

With basic whirlpool algorithm, there is a bottleneck for the last-hops. In order to deal with this bottleneck, we propose a modified algorithm.

```
BA(Sect[N], TotalDSize, RN )

INPUTS:
Sect[N] – array of N whirlpool sectors;
TotalDSize – total amount of data to be delivered from the network
RN – number of rotations;

BEGIN:
// Update data to be delivered from each sector in order
// to obtain required total amount of data

DSize[N]←DetermineDSize(Sect[N],TotalDSize),
NewSect[N] ← Update_data_size(Sect[N], DSize[N]),

// Identify and group non-conflicting sectors
NCGrp[M] ←Non_conf_sectors( NewSect[N]),

// Segregate Last Hops for each of the sectors and group
// together Last Hops for N sectors
LHGrp[N] ←SegLastHops( NewSect[N]),

FOR i←1 To M{
Sch[i]← Sched(NCGrp[M]);  }

FOR i←1 To N{
LHSch ← Last_Hop_Sched(LHGrp[N]); }

        // Start whirlpool rotations
      FOR  j←1  To  RN  {

       // Conducting one rotation
          FOR  j←1  To  M  {

            Execute_Sector_Group(Sch[M]); }

          Execute_Last_Hop(LHSch);  }

END
```

Figure 5. Basic Whirlpool Algorithm

4.2. OHT Algorithm: One Last Hop Through

In this algorithm we allow the last-hop for one of the sectors to occur concurrently while other sectors still deliver the data to the nodes before the last hop node. Similar to the basic algorithm sub-complete DTA schedules Sch[i] are generated for each of the whirlpool sectors. However, this time one last hop also scheduled in the sub-complete schedule. Consider again Figure 4. In case OHT the groups of non-conflicting sectors will be as follows:

Sch = [{o(c(t_{11} , t_{12}), t_{1lh}),c(t_{31} , t_{32})},

{o(c(t_{21} , t_{22}), t_{2lh}),c(t_{41} , t_{42})}]

```
OHT(Sect[N], TotalDSize, RN )

INPUTS:
Sect[N] – array of N whirlpool sectors;
TotalDSize – sizes of data samples delivered from each sector per rotation;
RN – number of rotations;

BEGIN:
// Update data to be delivered from each sector in order
// to obtain required total amount of data
DSize[N]←DetermineDSize(Sect[N]),
NewSect[N] ← UpdateDSize(Sect[N], DSize[N]),

// Identify and group non-conflicting sectors
NCGrp[M] ←NonConfSectors( NewSect[N]),

// Segregate Last Hops for each of the sectors and group
// together

// Last Hops for M sectors
LHGrp[M] ← SegLastHops(NCGrp[M]),

// Randomly select a last hop for each group
LastHop[M] ← RandomLastHops(LHGrp[M]),

FOR i←1 To M{
Sch[i]← Sched(NCGrp[M], LastHop[M]);  }

FOR i←1 To N-M{
 RemLHSch ← RemLastHopsSched(LHGrp[M]),
LastHop[M]; }

  // Start whirlpool rotations
        FOR  j←1  To  RN  {

         // Conducting one rotation
            FOR  j←1  To  M  {
              Execute_Sector_Group(Sch[M]); }
            Execute_Last_Hop(RemLHSch);  }
END
```

Figure 6. One-Hop Through (OHT)Algorithm

Each of the sector groups *Sch[1]* and *Sch[2]* are executed in the order of whirlpool rotation. Inter-sector concurrency is expressed as:

o(*a*(*o*(*c*(t_{11} , t_{12}), t_{1lh}), c(t_{31} , t_{32})),

a(o(c(t_{21} , t_{22}), t_{2lh}),*c*(t_{41} , t_{42}))). **(3)**

To complete the scheduling remaining two last hops are scheduled serially using DTA non-strict order operations::

LHSch = *c*(t_{3lh} , t_{4lh}) **(4)**

Combining schedules (3) and (4) as in the basic algorithm gives a complete schedule for one rotation:

$o\ (\ o(\ a(\mathrm{o}(\mathrm{c}(t_{11}, t_{12}), t_{1lh}), c(t_{31}, t_{32})\ ),$

$a(\mathrm{o}(\mathrm{c}(t_{21}, t_{22}), t_{2lh}), c(t_{41}, t_{42})\ )\ )\ ,\ c(t_{3lh}, t_{4lh})\ )\ ).$

4.3 Layered Whirlpool

In order to consider more concurrent transmission opportunities, we introduce layering in whirlpool. We consider how the whirlpool algorithms can be improved by introducing *interlayer concurrency*. First we explain Layered Basic Algorithm (LBA) where all last hops of the whirlpool are executed serially. Consider Figure 7a where the same color layer-sectors can be scheduled simultaneously. We assume that there is no conflict in transmissions across the layers. The outer-layer dark color sectors are scheduled initially to begin the whirlpool. The inner layer-sectors (closer to the base station) start transmitting only after the corresponding outer layer-sector transmissions have been completed. The layered whirlpool continuously acquires data from outer-layers and passes it through inner-layers to the center. Figures 7b, 7c and 7d represent three execution stages during one whirlpool rotation. All dark-color sectors at each stage can be executed simultaneously. After the stage 3 (Figure 7d), the last hops are executed serially. Figure 8 provides the LBA specification.

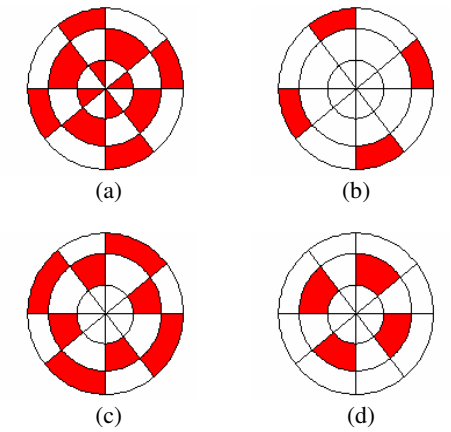

Figure 7. Explaining layer-sectored whirlpool execution

The second layered whirlpool strategy is the *LOHT* algorithm, which is the layered version of the OHT technique. The LOHT algorithm groups non-conflicting sectors and executes them along with one last hop from one of the sectors in the group. We also developed several advanced whirlpool algorithms maximizing the benefits of *continuous* concurrent transmissions. For example, Figure 9 illustrates the approach where transmissions from consequent whirlpool rotations co-occur. Figure 9d corresponds to the case where the last hop sectors are being executed while the outer-most layer starts executing transmissions for the second rotation. This maximizes the overall concurrency.

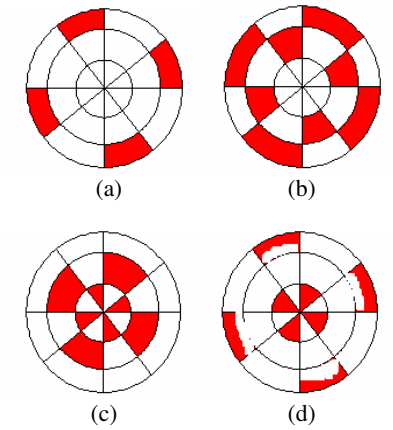

Figure 8. Whirlpool execution using advanced algorithm

PRECONDITION:
// Sensors area is layered and sectored (layer-sectored)

LBA(LSect[N], TotalDSize, RN)

INPUTS:
LSect[N] – array of *N* whirlpool layer-sectors;
TotalDSize – total data to be delivered from the network
RN – number of rotations;

BEGIN:
// Update data to be delivered from each sector in order
// to obtain required total amount of data

DSize[N]←**DetermineDSize**(*LSect[N],TotalDSize),*
NewSect[N] ← **Update_data_size**(*LSect[N], DSize[N]*),

// Identify and group non-conflicting layer-sectors
NCGrp[M] ←**Non_conf_sectors**(NewSect[N]),

// Segregate Last Hops for each of the layer-sectors and
// group together Last Hops for *N* sectors
LHGrp[N] ←**SegLastHops**(NewSect[N]),

// For each layer-sector *LSect[i] generate complete DTA*
// schedule *Sch[i]*
FOR i←1 ***To*** M{
Sch[i]← **Sched**(*NCGrp[M]*); }

FOR i←1 ***To*** N{
LHSch ← **Last_Hop_Sched**(*LHGrp[N]*); }

// Start whirlpool rotations
FOR *j*←1 ***To*** *RN* {

// Conducting one rotation
FOR *j*←1 ***To*** *M* **{**

Execute_Sector_Group(*Sch[M]*); }

Execute_Last_Hop(*LHSch);* }

END

Figure 9. Layered Basic Algorithm for whirlpool

5. EXPERIMENTS AND ANALYSIS

In this section, we provide an experimental evaluation of the whirlpool. We used a simulated sensor network with 75 sensors uniformly spread over a monitoring area. The whirlpool algorithms and other scheduling strategies were implemented in Arity Prolog 3.2.

5.1 Behavior of Whirlpool

In first set of experiments we evaluated the ability of whirlpool to utilize inter- and intra-sector (in-sector in Figure 10) concurrency. Here we used only basic whirlpool algorithm (BA). As a base line we used a serial strategy that executes transmissions in whirlpool sectors one by one. In addition, it schedules all transmissions within each sector serially. This corresponds to "No Inter-sector, No In-sector" in Figure 10. The next option is "Inter-sector, No In-sector" case, where whirlpool executes non-conflicting sector groups concurrently. Here, we can already see the benefits in response time when whirlpool is used. Then to take benefit from existing DTA framework, we apply DTA scheduling within each sector. The "In-sector, No Inter-sector" curve corresponds to DTA scheduling within each sector, while executing sectors one by one.

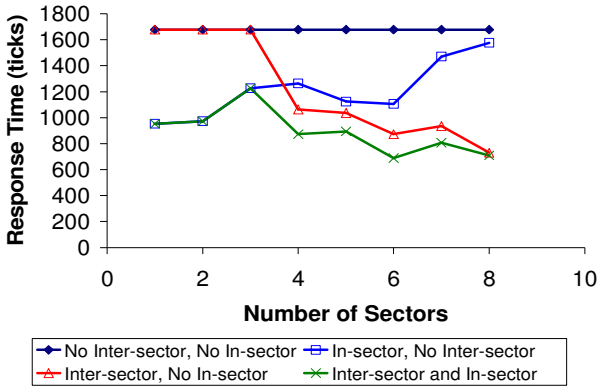

Figure 10. Effect of whirlpool concurrencies

Then, we combine inter-sector concurrency with DTA scheduling ("Inter-sector, In-sector" curve) in an attempt to maximize the concurrency benefits of DTA with basic whirlpool. Figure 10 shows that in general, any kind of whirlpool concurrency improves upon the base line. Inter-sector concurrency does not give any benefit for 1, 2 and 3 whirlpool sectors, since in this case any sector is adjacent with the rest of them and we do not have non-colliding sector groups. With an increase in the number of sectors the benefit of inter-sector grows, which is the expected behavior. Indeed, since execution time of a non-conflicting sector group is equal to the execution time of its maximal sector, the smaller sector sizes improve performance due to the benefits of inter-sector parallelism. Meanwhile, the benefit from DTA scheduling only ("In-sector, No-Inter-sector") degrades with a larger number of sectors. The reason is that smaller and narrower sectors provide fewer opportunities for concurrent transmissions. We observe that with 8 sectors DTA alone is approaching pure serial whirlpool execution, which means that only a few transmissions were scheduled concurrently. As expected, combining whirlpool and DTA scheduling demonstrates the best performance. In the cases of 1, 2, and 3 sectors the curve coincides with isolated DTA. With 8 sectors, where DTA performance is poor, the combined concurrency performance of basic whirlpool and DTA merges with the performance of isolated inter-sector parallelism. To sum up, inter-sector and in-sector concurrency considerably improves the whirlpool performance. The combined effect of both kinds of concurrency maximizes the whirlpool performance, which is consistent with our initial assumption and justifies our approach.

5.2 Comparison of Whirlpool Algorithms

Figure 11 compares overall response times for each of the whirlpool algorithms introduced in Section 4. Here we consider whirlpool with 3 layers. We observe a consistent improvement in the performance of the whirlpool algorithm comparing to the BA strategy. We also observe that the response time decreases as the serial bottleneck for the last-hops is rectified. Meanwhile, most considerable impact on response time comes from the layering. The LOHT strategy wins over all other algorithms for almost all number of the whirlpool sectors. The only exceptions are 1 and 2. In this case the performance of LOHT degrades since the execution time of the additional last hop becomes comparable with the execution time of a next layered-sector. For 3 or more sectors, LOHT behaves better than other algorithms, since in this case it considerably eliminates the last-hop bottleneck. However, with the increase in the number of layer-sectors the layering effect is diminished since the smaller size of the layer-sectors restricts intra-sector concurrency. It should also be noted that the complexity of the LOHT scheduling may enforce choosing simpler but less efficient algorithms such as LBA and OHT.

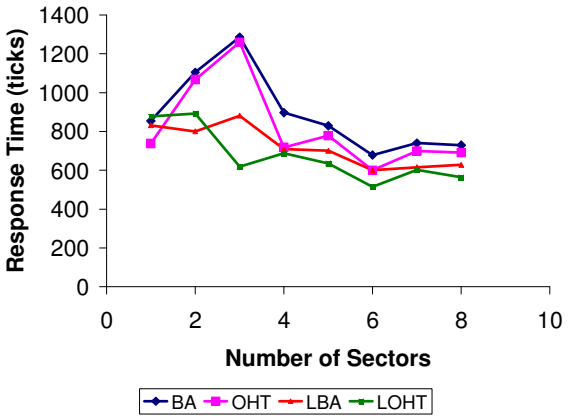

Figure 11. Response Time for whirlpool algorithms

In order to investigate the utility of whirlpool for timely data delivery, we evaluated the data delivery profiles for each of the whirlpool algorithms. Figure 12 presents the results of this evaluation for the whirlpool structure with 8 sectors and 3 layers. We observe that OHT algorithm starts delivering data much earlier than BA. Similar relationship exists between LOHT and LBA. As expected the both LBA and LOHT over perform BA and OHT. The ability of whirlpool to deliver data early is very useful for mission-critical system monitoring. The crossover in the data delivery profiles indicates performance trade-offs that should be explored by our system. For example, we may choose OHT over LOHT if the early data delivery is preferable. Meanwhile, if faster overall response time is required then LBA should be chosen.

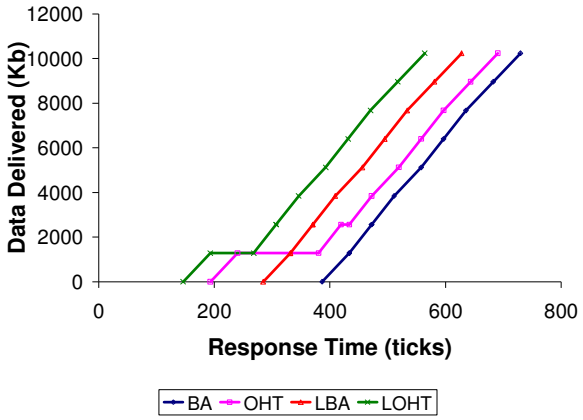

Figure 12. Data delivery profile for whirlpool algorithms

Figure 13 illustrates further possibility of improvement in the whirlpool algorithms via better time synchronizations. We compare the LOHT strategy with the two advanced algorithms that perform synchronized whirlpool interrogation using the idea outlined in Figure 9. Adv1 and Adv2 are improvements over LBA and LOHT algorithms respectively. Figure 13 represents benefits from one rotation only. Meanwhile the performance gain will increase with increase in number of rotations, since some of the outer-layer transmissions can be executed concurrently with the last hops of the previous whirlpool rotations. Although it is beneficial yet, the time-synchronization is difficult to implement. We are currently exploring the practical applicability of these algorithms.

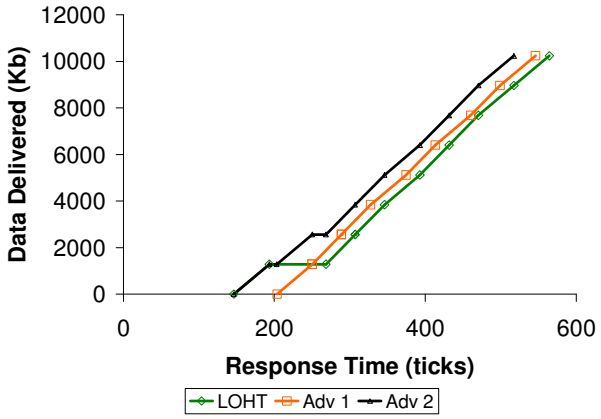

Figure 13. Profile for advanced whirlpool algorithms

To summarize, our experiments demonstrated high utility of the whirlpool-based strategies for timely data delivery in sensor networks. We also demonstrated how whirlpool algorithm can be tuned to maximize the transmission concurrency in order to meet the given performance target. In related research, we demonstrated that whirlpool can be efficiently used to implement a monitoring system for early detection of instability in complex structures [6].

6. RELATED WORK

Data dissemination schemes like SPIN [7] using flooding technique, interest gradient based Directed Diffusion [8], clustering based LEACH [9], GAF [10] have been proposed in literature. Wave scheduling [11] minimizes packets collisions by carefully scheduling the sensor nodes. It results in energy savings at the expense of increased message latency. Synopsis Diffusion [12] proposes a multi-path routing scheme which is more robust than tree topology based TAG [13] to avoid double-counting in sensor readings. In related research [2, 3], we are also developing *cross-layer optimization* strategies using DTA and 802.15.4 protocol, which we believe will be energy efficient and along with *whirlpool* will develop into a robust, scalable, concurrency-intensive wireless sensors data delivery system that will deliver data reliably and respond to queries with low latency.

7. CONCLUSION AND FUTURE WORK

In this paper, we introduced a *rotating interrogation* technique called *Whirlpool* that provides opportunities to optimize data delivery in time-critical monitoring applications. We explored concurrency opportunities with different whirlpool algorithms. We also demonstrated extra benefits of using whirlpool in combination with our algebraic query optimization framework.

In future work, we plan to study energy considerations and test sleep-wake strategies along with whirlpool and Data Transmission Algebra. We will develop robust strategies to deal with network failures based on utilization of near-by sectors for inferring the relevant data. Currently we are conducting further study of the whirlpool for mission critical applications such as Structural Health Monitoring [6].

8. ACKNOWLEDGMENTS

We would like to thank Prashant Krishnamurthy, Alexandros Labrinidis and Daniel Mosse for their valuable feedback.

9. REFERENCES

[1] Xu, N., Rangwala, S., Chintalapudi, K.K., Ganesan, K., Broad, A., Govindan, R., and Estrin, D. A wireless sensor network for structural monitoring. SenSys 2004.

[2] Zadorozhny, V., Chrysanthis, P.K., Krishnamurthy , P. A Framework for Extending the Synergy between MAC Layer and Query Optimization in Sensor Networks, DMSN, 2004.

[3] Zadorozhny, V., Sharma, D., Krishnamurthy, P., Labrinidis, A. Tuning query performance in sensor databases, MDM, 2005.

[4] IEEE Standard for Information technology - Local and metropolitan area networks: Part 11: Wireless Local Area Network Medium Access Control (MAC) and Physical Layer (PHY) Specifications, June 1997.

[5] IEEE Standard for Information technology - Local and metropolitan area networks: Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), October 2003.

[6] Sharma, D., Zadorozhny, V., Chrysanthis, P.K. Structural Health Monitoring with Whirlpool, to appear in IWSHM 05.

[7] Heinzelman, W., Kulik, J., and Balakrishnan, H. Adaptive protocols for information dissemination in wireless sensor networks, in MOBICOM 1999.

[8] Estrin, D., Govindan, R., Heidemann, J., and Kumar, S. Next Century Challenges: Scalable coordination in sensor networks, in MOBICOM, 1999

[9] Heinzelman, W., Chandrakasan, A., and Balakrishnan, H., Energy-efficient communication protocols for Wireless Microsensor Networks, Proc. Hawaaian Int'l Conf. on Systems Science 2000.

[10] Xu, Y., Heidemann, J., Estrin, D. Geography-informed energy conservation for ad hoc routing, in MOBICOM, 2001.

[11] Trigoni, N., Yao, Y., Demers, A., and Gehrke, J. Wave Scheduling: Energy-efficient data dissemination for sensor networks, in DMSN, 2004.

[12] Nath, S., Gibbons, P.B., Seshan, S., Anderson, Z.R. Synopsis diffusion for robust aggregation in sensor networks, in SenSys, 2004

[13] Madden, S., Franklin, M.J., Hellerstein, J.M., and Hong,W. TAG: A Tiny Aggregation service for Ad-hoc sensor networks, OSDI 2002.