IQ2s: Workshop on Information Quality and Quality of Service for Pervasive Computing, Lugano, Switzerland, March 19, 2012, Proc. 2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), IEEE Press, 2012, pp. 14-19.

Transient Traffic Congestion Control with Traveling Auctions

S. Yousaf Shah and Boleslaw K. Szymanski
Department of Computer Science and Network Science and Technology Center
Rensselaer Polytechnic Institute (RPI), Troy, NY, USA
Email: {shahs9,szymansk}@cs.rpi.edu

***Abstract*—The use of market mechanisms to solve computer science problems such as resource sharing, load distribution and network routing, is gaining significant traction. In this paper, we investigate new market mechanisms to solve the problem of bandwidth sharing in wireless networks for transient traffic congestion often generated by event-driven packet flows. Typically, such congestion is transient at each node it arises since the bursts of data move following the event. We first demonstrate that a previously proposed strategy that greedily selects winners in repeated routing auctions is not globally optimal in such a case. We also demonstrate, by evaluating a *lookahead* mechanism for winner selection in the corresponding auctions, that greedy algorithm approximates optimal selection very closely. Then, we introduce and evaluate a novel mechanism which we call *Traveling Auctions* to address the problem of transient congestion. We experimentally show that using *Traveling Auctions* mechanism improves the network performance.**

***Keywords*-Wireless Sensor Networks (WSNs); Quality of Information (QoI); Auctions; Congestion**

I. Introduction

Typically congestion control in communication networks is based on a feedback loop from congested nodes to the sources that limits the source transmission rates, when appropriate. Since the feedback loop takes time to adjust the rates, this approach works well only for stable flows. However, in many situations, congestion may arise from burst of packets triggered by mobile events in the environment. These events, and often the spot at which packets are generated, move as the event evolves. A typical scenario may involve traffic generated by people and the devices operating during the rescue operation in a disaster stricken area. In such a case, the packets sent by them may be prioritized by the urgency assigned to the message that they carry. Moreover, they can travel along the regular or ad hoc deployed wireless network.

Congestion delays packets on their way to the destination, which lowers QoI provided by the corresponding application. Each application has a specific QoI function whose one of the arguments is the delay of the information received at the destination since its inception at the source. In this paper, we use a simple QoI function that defines the loss of QoI resulting from the delay of the packet during transmission from the source to the destination as the product of this delay and the packet priority. Our approach applies directly to any application in which loss of QoI is a linear function of the packet delay, and requires just straightforward modifications for application in which the loss is monotonically non-decreasing function of the delay.

In [1], the authors propose an auction mechanism for resource allocation in WSNs. Whenever congestion occurs on a node, it acts as an auctioneer and runs an auction for the available transmission slot. Each packet bids its predicted QoI loss that would result from not receiving the next available transmission slot. By choosing bid values, the auction can either minimize the total QoI loss for all applications, or equalize the QoI loss among all applications. In auction mechanism proposed in [1], the packet with highest bid or in other words with highest QoI loss wins auction and is transmitted. A receiver-assisted congestion control (RACC) mechanism, based on TCP is proposed in [2]. In RACC the receiver estimates transmission rate based on packet inter-arrival time at the receiver-end and notifies the sender of the rate so that the sender adopts the rate to avoid congestion. Buffer based mechanism to avoid congestion has been proposed in [3]. In which each node maintains a buffer state which is piggybacked to its neighbors. A node forwards packet to the other node only if its buffer status is nonfull, this way the whole network adopts to a rate that would not create congestion. In [4], priority based congestion control (PCCP) is proposed. PCCP protocol detects congestion using its intelligent congestion detection (ICD) mechanism and piggybacks implicit congestion notifications in the header of data packets. ICD uses both packet inter-arrival time as well as packet service times to detect congestion. PCCP also has priority based rate adjustment (PRA) mechanism in place in which each sensor node is given priority index, and bandwidth assigned to nodes is proportional to their priority index.

Above mentioned approaches (except [1]) use feedback loops in one way or the other, some of them are also rate limiting protocols. As explained earlier, such mechanisms for congestion control are not feasible for event driven traffic in WSN. Our approach, like [1], uses auctions to resolve transient congestion at each node. Previously considered approaches of this kind [1], considered each node in isolation from others, and their greedy winner selection was optimal if very long (theoretically infinite) streams of packets were congested. In this paper, we show that in the case of

limited length of packet streams, a lookahead based winner selection can improve the results slightly, thanks to its ability to accelerate gains as early as possible. At the same time, the cost of lookahead rises quickly with the horizon (number of future auctions considered) of lookahead and its benefits decrease as the number of packets competing for transmission in a congested node increases. Hence, our results demonstrate that the greedy winner selection is an excellent and simple approximation of the optimal selection.

To further address the challenges of transient congestion resolution, we also introduce a traveling auction in which its state at the currently congested node is piggybacked in the transmission packets. Consequently, each neighbor of the congested node can continue the current auction in case congestion moves there. The main contribution of this paper is a solution to the transient congestion problem arising in wireless networks when the sources of streams of packets move from node to node shifting their trajectories and the congestion points of the traffic. We developed a novel *Traveling Auction* mechanism in which the state of auction execution transfers from node to node following the congestion.

II. Motivating Scenarios

As explained in the previous section, our approach is based on traveling auction and applies to scenarios in which congestion moves from node to node. Some of these scenarios include (i) object tracking, (ii) unmanned vehicles performing joint operation and communicating over wireless network, and (iii) rescue workers using an ad-hoc wireless network in a disaster stricken area exchanging messages of various priorities with aid agencies, hospitals and general public. All the aforementioned scenarios involve mobile objects with prioritized data traffic in which an efficient congestion resolution mechanism is needed to effectively use the WSN bandwidth. In this paper, we demonstrate our approach benefits in the scenario of a tracking application.

Consider a car chased by police; as the car moves along its path in the sensor network, the sensor nodes sensing the car generate bursts of data. As the followed and following cars move forward, they enter into the sensing range of new sensor nodes and exit the range of sensor nodes behind them. Consequently, the bursts of packets are generated by different sensor nodes as the cars continue the chase. These bursts generate congestion on the routing nodes, with the congestion moving from one routing node to another as the bursts change their origins and trajectories. These bursts are of short duration but produce intensive traffic.

In previously developed solutions, a static auction at a congested node starts from scratch when the congestion arises and does not take into account the times at which the last packets of each application passed to the destination. This makes the first round of sending packets round robin as each application sends one packet out in the order of its priority. Thus, packets with lower priorities may wait shorter or longer for winning compared to the case in which the congestion would stay at the previous node.

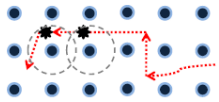

Figure 1. Congestion traveling along with the object

III. Auctions for congestion control

Auctions have proven to be useful in distributed resource allocation. Auctions were used in [1] for transient congestion. They use greedy winner selection in which the winner of each auction is the packet with the highest QoI loss. Thus, the greedy strategy considers the QoI loss only in the current auction. However, when the streams of packets creating a congestion in the given node are short, the greedy selection may not be optimal.

We developed and evaluated *lookahead* mechanism for winner selection in which we consider a sequence of auctions, not just one, and find a sequence of winners that reduce the QoI loss the most. Here is our motivation for investigating the $lookahead$ mechanism. Consider just two applications, a_1, a_2 with single auction QoI loss of $s_1 > s_2$ and the total accumulated loss at the current auction $l_1 < l_2$, respectively. If $s_1 + l_1 > s_2 + l_2$, the greedy selection of a winner in each auction yields a_1, a_2 and saves $2l_1 + l_2 + s_1 + s_2$ of QoI loss, while the opposite order of winners saves $l_1 + s_1 + s_2 + 2l_2$ of QoI loss. Yet, the latter is larger than the former. This happens when a low priority packet accumulates QoI loss just slightly above the loss of a high priority packet. In general, *Lookahead Strategy with Horizon h* finds the best sequence of winners in the current and h future auctions and selects the first application in this sequence as the winner of the current auction.

In this paper, the goal of an auctioneer is to minimize the total QoI loss for all the applications. Let p_i be the priority of the application a_i. This priority is a function of the subjective measure of importance of the application and objective measure of the content of the packet (e.g., in our application that could be the speed of the object, direction of its movement, etc.). We assume the TDMA MAC layer protocol in our scenario with TDMA slot being large enough to transmit one packet every time unit independently of the time of the auction, but in general this may not be the case. tmi denotes the time at which the packet of application i currently competing in the auction was generated (in our applications, this is the measurement time of the object). $tmpi$ is the packet origination time of the last packet for application i that was transmitted by the node. Since we assumed that the QoI loss caused by losing the next auction is a linear function of the delay, the bid b_i for the application

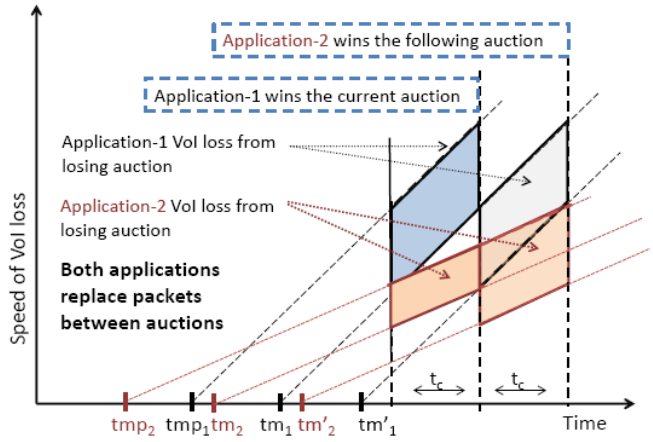

Figure 2. Sink replacing an old packet with a fresher one; tmp_1 and tmp_2 are either the origination times or zeros whereas tm_1 and tm_2 are the origination times of the packets of applications 1and 2. respectively.

a_i represents this loss, so it is equal to:

$$b_i = (tmi - tmpi)p_i t_c \tag{1}$$

If the QoI loss is a non-linear function of the packet delay, the bid computation will be just a difference between the predicted QoI loss with winning the current auction versus this loss with losing the current auction and winning the next one. Therefore, packets bidding in the auction calculate their bids based on above mentioned parameters, that are known at each node including priority (carried in packet) which is specific to source application and highly impacts the bids. In short, each packet contributes two application specific factors to the formation of the bid, the application priority and lateness of the packet.

From the above definition of QoI loss, it is clear that when a new packet of the application for which we already have a packet in the queue waiting to be delivered arrives, only the packet with the latest origination time needs to be stored and the older one can be discarded. When a new packet replaces the old one for the same application, its bid grows too because the $(tmi - tmpi)$ increases. So, the more current information sink has about the object the lower is the QoI loss caused by delays. We understand that this replacement of packets creates loss as we are skipping a packet but as we can see from the figure (2) it is optimal decision at this point. Figure (2) illustrates the QoI losses saved when the latest available packet is delivered.

For lookahead with a horizon h, we need to calculate h future bids for each application in addition to the current bid. Winning the auction changes value of $tmpi$ for the subsequent auctions, so the QoI loss of application a_i, and therefore its bid in future auctions depends who was the winner in previous auction. Moreover, it is also important to know if the winning application will participate in the future auctions at all, which will not be the case if the application shifts its path and no longer routes the packets via the current node. Even if the path continues through the current node, in general the node cannot know when it will receive next packet for this application and what origination time that packet will carry. This means we need a mechanism to predict the next possible tmi. In order to predict the future tmi, we employed two techniques. The first one is based on recent history and it computes the predicted origination time of the future packet as:

$$tmi_{t+k} = tmi_t + k(tmi_t - tmi_{t-1}) \tag{2}$$

where tmi_{t-1} is the origination time of the packet of application i just before the current packet was received at the node. Equation (2) calculates the next possible tmi_{t+k} for application i by adding previous inter-packet generation delay observed at the node up to the current packet generation time. The other technique is based on packets received per transmission slot and the computation in this case is:

$$tmi_{t+k} = tmi_t + kt_c \frac{Nm_i}{N_{slots}} \tag{3}$$

Nm_i is the number of packets received from this application in the given time, and N_{slots} is the number of transmission slots that arrived at the node during this time. Equation (3) calculates the next possible tmi_{t+k} for application i by adding average packets per slot to the current packet generation time.

Our experiments show that history based approach gives better results. The future bids are dependent on who is the winner of each preceding auction. As we proved in [5], by sorting all possible bids for the current auction, only h highest bids can be potential winners for the first auction in the sequence of h future auctions. Still, the number of alternative winners in the auction followed by at most h future auctions is limited only by 2^h. Hence, the overhead of lookahead with horizon h grows very quickly with h.

In the static auction discussed so far, each congested node does not take into account the auction information on the other nodes. Thus, every node when it becomes congested, starts an auction based only on its local information. To address this shortcoming, we developed traveling auctions strategy in which the auction history travels along with the congestion.

As discussed before, it is just the $tmpi$ value that defines the current QoI loss of the application, so to enable sharing this value with the neighbors of the congested node, this value is added to the header of each winner packet being transmitted for the node. As a result, if the congestion and consequently also the auction move to the neighboring node, the node will be able to compute losses of all applications correctly. Sharing of the auction state information is done via each node overhearing its neighbors. When a node forwards a packet to another node, all the neighbors of the sender

node listen to it, and read the packet's header to check the destination address of the packet against their own addresses; nodes or a node to which the packet is forwarded receive it (as the destination address of the packet and node's address matches), and rest of the nodes extracts the application id and the $tmpi$ of that packet, note that this is tmi of the packet, it becomes $tmpi$ for subsequent packets. It is important to notice that this is not data level overhearing but it leverages MAC level filtering to share the auction state information which is less costly than application level data overhearing. For our simulations, we have extended the packet header in *NS2* to include this additional information there.

IV. Performance Evaluation

A. Testbed

To evaluate 'Lookahead' with horizon up to two, and 'Traveling Auctions' approaches, we implemented the object tracking scenario using the Network Simulator *NS2*. We simulated a sensor network of 44 nodes with number of mobile objects (nodes) varying between 2 and 4, each of which was served by a separate application of the object tracking algorithm. The sensing nodes were positioned at fixed points, while the tracked objects were moving according to certain mobility patterns to mimic real life objects moving in a sensor network while being tracked by the sensing nodes. We used *NS2* native DSDV as routing protocol that routes packets via the dynamically chosen shortest path to the destination.

We have developed a traffic generator that is attached to all sensing nodes. It generates customized UDP packets with extended header containing scenario specific information, such as the total number of applications, the associated application id, the time of the packet creation, and the application priority. Whenever the object enters the sensing range of a node, the node starts generating object measurement packets for the application associated with this object at a rate of 10 packets per second.

B. Evaluating Lookahead vs. Greedy Selection

We simulated three different mobility models: "Random Waypoint Model (RWM)", "Pursue Mobility Model (PMM)" [6] and "Manhattan Grid Model (MGM)". We generated the mobility traces using BonnMotion tool [7]. We experimented with these mobility models over the diverse set of priorities, and with the number of objects varying from 2 to 4. We set packet generation rate at the sensor node high enough to create congestion somewhere downstream. The congested nodes then run auction to find the winner for the next transmission slot. The winner is selected in three different ways: Greedy, Lookahead with $h = 1$ (LA-1) and $h = 2$ (LA-2). We ran simulation for 200 seconds and recorded the sum of QoI losses for all applications at the sink.

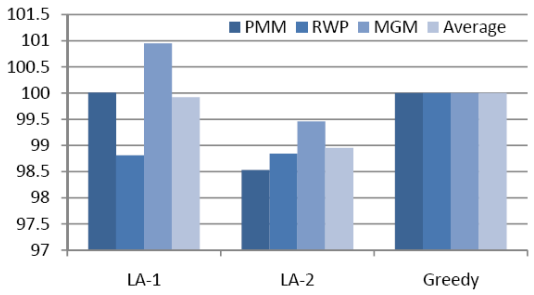

Figure 3. Performance of Lookahead measured as percentage of greedy performance under different mobility models

Table I presents the average QoI loss for all applications under different mobility models for all objects.

Table I
AVERAGE QOI LOSS FOR APPLICATIONS UNDER DIFFERENT MOBILITY MODELS.[1]

Objects	LA-1	LA-2	Greedy	Runs	Mobility Model
2	1330	1357	1331	7	PMM
3	5247	5332	5267	7	PMM
4	7622	7683	7592	7	PMM
2	2029	2021	1982	7	RWP
3	5309	5403	5344	7	RWP
4	9095	8965	8933	7	RWP
2	2347	2388	2378	7	MGM
3	5119	5277	5135	7	MGM
4	8691	8662	8802	7	MGM

Figure 3 presents the average performance of lookahead winner selection normalized to the greedy winner selection under different mobility models. Since there is either very little or no improvement from using lookahead versus the greedy selection, we use the latter in *Traveling Auctions* mechanism.

It is obvious from the results that there is no single strategy which is optimal for all cases. Since the sensing nodes generate packets only when they sense an object, the generated traffic behavior is unpredictable. The results show that by increasing the lookahead horizon, in some cases, we can slightly improve the results. Yet, such slight improvement triggers increase of computational overhead incurred by lookaheads with larger horizons. Since continuation of the current auction at the given node is never guaranteed (as the congestion may move to another node at any time), the added overhead may not improve the results. If the auctions are running very long, LA-1 is better in terms of the avoided QoI loss, but it requires the investment (non-optimal step from the perspective of a single auction) which may be lost if the traffic moves. We have developed a simulator in JAVA

[1]Priority sets: {(5,10), (4,16), (4,8), (9,36), (2,8), (3,6),(7,14)}, {(5,10,15), (4,16,36), (4,8,12), (9,36,81), (2,8,27), (3,6,9),(7,14,21)}, {(5,10,15,20), (4,16,36,64), (4,8,12,16), (9,36,81,144), (2,8,27,64), (3,6,9,12),(7,14,21,28)}

to simulate the long running auctions and verify that LA-1 is better in such situations. Whereas if there are fewer auctions or stream of packets is finite, then simple greedy outperforms lookahead mechanism. For the sake of space these results are omitted here, but available at [5]. Luštrek and Bulitko have also found in [8] that lookahead with the unit horizon is the best for the real time path finding problems.

C. Evaluating Traveling Auction

We evaluated the *Traveling Auction* mechanism with PMM and compared it with static greedy auction mechanism under the same mobility model[2].

Table II
CONFIGURATION PARAMETERS FOR TRAVELING AUCTIONS

Objects	Bandwidth	Packet Rate	P.G	P.Rec
2	200kbps	10 pkts/sec	5403	1304
3	200kbps	10 pkts/sec	8111	915
4	200kbps	10 pkts/sec	10801	947

Table II shows parameters used in experiments. The columns 'P.G' and 'P.rec' represent the total number of packets generated and received, respectively. Since the objects move from outside of the sensing range, they initially travel through sparse portion of the field, and the number of packets generated reflects that. In case of three applications, the overlap of sensing ranges of nodes causes on average 1.36 packet generated for each objected sensed. Hence, with 200 seconds of simulation and 3 objects in range, there would be 8160 packets, while actual number is 8111. Packets received arrive on average with 200/915 =0.219 sec delay. By design, packets leave each node every 0.1sec. so the waiting time for arrival of the transmission slot at the node ranges from 0 to 0.0975 with each transmission, so average delay is 0.050674. The average number of hops recorded was four and therefore delay is about 0.203 which is consistent with the delay observed. We observed the same consistency of expected and recorded results in the remaining cases of two and four applications.

The figure 4 shows QoI loss under comparative strategies. The no congestion, referred here as *network losses*, are losses accumulated by all the applications because of non-congested traffic delays and discontinuous measurements of the target object movements. Therefore, the total QoI losses caused by: (i) Non-continuous measurements: Depending upon the frequency of measurement of the position of the object, the application accumulates loss between two subsequent measurements. Higher frequency of measurement would lower the loss and vice versa. (ii) Non-congested traffic delays: During the time when a packet traverses its path from source to sink even under the non-congested

[2]We also experimented with lookahead as the winner selection strategy for traveling auction but the greedy selection yielded better results.

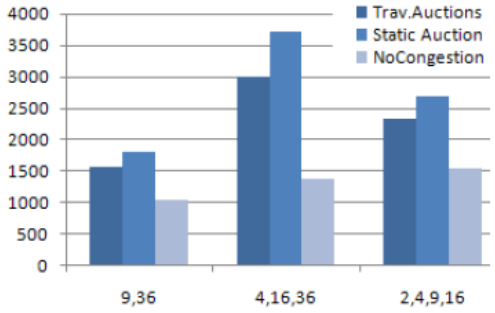

Figure 4. QoI losses for different set of applications

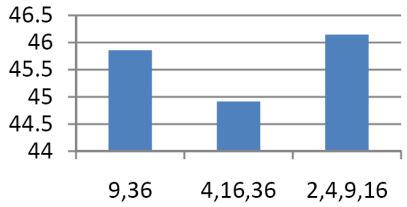

Figure 5. Percentage of Traveling Auctions Gain for various sets of priorities

network, the object measurement information carried by a packet becomes outdated by the time this packet reaches the sink. Longer the path from the source to destination, larger is the loss incurred by the application. (iii) Congestion delay: While there is congestion in the network, some of the packets wait at the congested routing nodes as the upcoming transmission slots are taken by the packets of other applications. We can reduce this loss and thus the total QoI loss by efficiently managing the order in which packets of various applications are assigned incoming transmission slots.

The network losses for two applications with priorities (9,36) are shown in the figure 4. During the simulation, the sink received 2000 packets. The positions of objects were measured every 0.005s, so loss caused by the non-continuous measurements was $0.005/2 * 2000 * (36 + 9) = 225$. Hence, the QoI loss due to the non-congested traffic delay is $1033 - 225 = 808$ and this bounds the delay that was at least (1033-225)/2000/(36+9) = 0.009s and at most (1033-225)/2000/(36+9)*2=0.018s. The delay measured in the simulation was 0.012 sec., so consistent with the bounds. In case of three applications with priorities (4,16,36), the minimum cost that is caused by non-continuous measurements is $0.005/2 * 2000 * (4 + 16 + 36) = 280$ and caused by the non-congested traffic delay is $1359 - 280 = 1079$. The corresponding bounds on network delay are $1079/2000/(4+16+36) = 0.0096sec$ and $1079/2000/(4+16+36) * 2 = 0.019sec$ which again is consistent with the measured delay of 0.012 sec. The same consistency was observed for four applications.

Auctions impact only congestion caused losses of QoI,

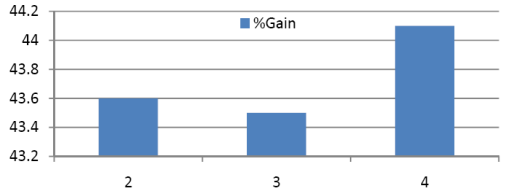

Figure 6. Average gains for different set of applications.[3]

so we measured how much higher were the Static Auction QoI loss relative to Traveling Auction VoI loss using the following formula,

$$Gain = 100 * \frac{LSA - LTA}{LTA - Loss\_Network} \quad (4)$$

where LSA is the QoI loss in Static Auctions while LTA is the QoI loss in Traveling Auctions.

As shown in figure 5, Static Auctions incur 45% higher losses relative to Traveling Auctions. In order to affirm the scale of performance gains, we conducted tests for various priority sets for 2,3 and 4 mobile objects under the same setup. Figure 6 shows average of performance improvements for each number of mobile objects tested. The actual values of losses are not presented here in the interest of space, instead we computed the improvements for each individual case and then took their average.

For optimal sharing of bandwidth in case of traveling congestion, sensor nodes need to share information on auctions that are held at the neighboring nodes to provide continuity in congestion resolution. As we have shown, in such highly bursty traffic situations $StaticAuctions$ performs on average up to 45% worst than the $TravelingAuctions$. However, if the network is dense relative to the node's sensing range (so there are many neighbors in the sensing range of an average node), then static auctions perform better.

V. Conclusion

In this paper, we have investigated *lookahead* mechanism for winner selection in auctions that are used for congestion resolution in wireless networks. We have compared traditional greedy winner selection mechanism with lookahead mechanisms introduced in this paper. We have also proposed the new *Traveling Auctions* mechanism, in which the state of the auction moves along with the congestion in the network in order to minimize the total QoI loss. The performance results show that the Traveling Auction improves routing decisions and reduces the QoI loss in scenarios in which high bursts of event driven data trigger intense but transient congestion in wireless sensor networks. For future research, we plan to investigate market based routing in which low priority packets based on expected routing delays would dynamically choose alternative but longer paths to the destination. This will enable the low priority packets to avoid competing with high priority packets on congested nodes for the shortest paths and possibly reduce delivery delays.

[3]Priorities: {(9,36), (2,8), (7,14), (6,12), (3,6), (4,16), (5,10)},{(5,10,15), (2,4,9), (3,6,9), (7,14,21), (6,12,18), (4,16,36), (4,8,12)},{(5,10,15,20), (2,4,9,16), (7,14,21,28), (6,12,18,24), (3,6,9,12), (4,16,36,64), (4,8,12,16)}

VI. Acknowledgment

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] L. Chen, Z. Wang, B. Szymanski, J. Branch, D. Verma, R. Damarla, and J. Ibbotson, "Dynamic service execution in sensor networks," *The Computer Journal*, vol. 53, no. 5, p. 513, 2010.

[2] K. Shi, Y. Shu, O. Yang, and J. Luo, "Receiver-assisted congestion control to achieve high throughput in lossy wireless networks," *Nuclear Science, IEEE Transactions on*, vol. 57, no. 2, pp. 491–496, 2010.

[3] S. Chen and N. Yang, "Congestion avoidance based on lightweight buffer management in sensor networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 17, no. 9, pp. 934–946, 2006.

[4] C. Wang, B. Li, K. Sohraby, M. Daneshmand, and Y. Hu, "Upstream congestion control in wireless sensor networks through cross-layer optimization," *Selected Areas in Communications, IEEE Journal on*, vol. 25, no. 4, pp. 786–795, 2007.

[5] S. Y. Shah, "Use of auctions in wireless sensor networks," Master's thesis, RPI, 2011. [Online]. Available: http://www.cs.rpi.edu/∼szymansk/theses/shah.ms.11.pdf [Oct,10,2011]

[6] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad-hoc network research," *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 483–502, 2002.

[7] N. Aschenbruck, R. Ernst, E. Gerhards-Padilla, and M. Schwamborn, "Bonnmotion: A mobility scenario generation and analysis tool," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010, pp. 1–10.

[8] M. Luštrek and V. Bulitko, "Lookahead pathology in real-time path-finding," in *Proceedings of the National Conference on Artificial Intelligence (AAAI), Workshop on Learning For Search*, 2006, pp. 108–114.