SIMTorrent: Sensor IMage Torrent

Anirban Banerjee
Department of Computer Science
University of California, Riverside
Email: anirban@cs.ucr.edu

WeeSan Lee
Department of Computer Science
University of California, Riverside
Email: weesan@cs.ucr.edu

***Abstract*— Network Reprogramming of sensor nodes is a necessity, with the explosion in the number of individual sensors being deployed for live experiments as well as in testbeds. Personalized physical reprogramming of large number of nodes is impractical both in terms of manpower cost and the amount of time expended for such purposes. We propose a network reprogramming protocol specifically targeted at deployments of heterogeneous varieties of sensor platforms. Our protocol efficiently disseminates specific program images to the targeted group of nodes for reprogramming. We achieve performance improvements over Deluge, and quantify other benefits achieved by employing our protocol. Current sensor image dissemination protocols fail to handle deployments consisting of more than one kind of sensor platforms, while the use of our protocol allows for deployments consisting of heterogeneous mix of sensor platforms.**

I. Introduction

Simply put, wireless sensor networks (WSNs) are clusters of small-embedded wireless devices cooperating on one or a small number of sensing and reporting based tasks [1], [2]. Some of the most crucial design aspects for WSNs are power frugal operation [3] and unmanned deployment and autonomous operational capabilities [4]. Sensors networks today employ platforms which sport a plethora of sensing devices, temperature, pressure, humidity, light, carbon dioxide and others [3]. Deployments of these multifaceted sensing devices is usually meant to gather ambient information about the environment. Network Reprogramming [5] provides us with the capability to change completely the program image executing on the small sensing devices. This is imperative if a large cluster of sensor needs to change its sensing paradigm, parameters, or load a new image which will allow it to adapt to the changing needs of the end user. Reprogramming nodes physically is impractical once deployment has been activated, furthermore with increasing testbed sizes in research organizations and universities there is a definite need to develop an efficient bulk data dissemination protocol. Moreover this protocol must be able to successfully program remotely, heterogeneous clusters of nodes. Such ability allows for network designers to employ different nodes, with different sensing and processing capabilities, possibly made by different manufacturers to cooperate with each other and perform the needed sensing tasks seamlessly. A network reprogramming protocol, which can handle heterogeneous clusters of sensor nodes allows for gradual deployment of different varieties of nodes, often needed as the sensing needs change, consider, for example, a stage by stage deployment of sensors specifically built to log data about ecological parameters, followed by highly specialized sensor only built for logging voice data [3]. These two types of sensors may not share the same platform, due to the obvious difference in the processing capabilities needed in the two tasks. Harmonic extraction and cepstral analysis of voice data requires at least 4,000,000 Samples per second, well out of scope of 8051 based MCUs and ATMEL128L based Mica's [3].

The network programming protocol must be able to reprogram over the wireless interface, the specific types of nodes for which the images are pumped into the network. These images can be and are in most cases larger than the RAM sizes, typically 4K or 8K, thereby there must be an intelligent methodology which is able to fragment pages to be distributed into packets; which we term as cells. Fragmentation sizes for the packets may not be the same, the reason being that the wireless links between nodes of heterogeneous varieties may have different capabilities in terms of peak data rate, modulation schemes, BER and others. Thus for each pair of nodes, of different type there needs to be a generic method to determine the best configuration for cell sizes depending on the link parameters. Additionally issues relating to suppression of image request messages and their aggregation are critical from a performance point of view. SIMTorrent address these issues and more in order to come up with a scalable, efficient and simple to implement bulk data dissemination protocol.

This paper provides two main contributions. First, we present SIMTorrent, a reliable data dissemination protocol for heterogeneous sensor network deployments, specifically targeted at distributing larger than RAM images, from one or more source nodes to many different types of nodes in a multi-hop sensor network. SIMTorrent is cluster aware; meaning that it attempts to identify clusters of nodes which need a particular image and cuts down on the requests for particular pages using either aggregation or suppression as configured by the network designer. SIMTorrent provides for hop-by-hop reliable data transmission by employing TCP like, exponentially increasing, ACK message windows, using a 16 bit CRC as the trigger for identifying any bad cell reception. Second, we simulate and present results of our protocol, to highlight the benefits achieved by implementing it in a simulated sensor network.

In Section II we discuss extensively the related work that has been part of this exciting research area. Followed by the detailed Protocol overview and description in Section III.

Section IV enumerates our experimental results which conclusively prove the efficacy of our protocol, followed by Section V, detailing out our intentions for extending our efforts in order to include even more functionality in the protocol specifications. Finally rounded up with conclusions in Section VI.

II. Related Work

The problem that we address in our research pivots on reliable dissemination of data to a heterogeneous collection of sensor nodes. The main highlights of our work, include, a scheduling policy allowing for a common datum based approach in order to reduce the number of transmissions required for the generic ADVERTISE-REQ-RECEIVE handshake mechanism. Furthermore we develop a gradient tree based approach to get information from the sink to the nodes for which the cells, of a particular image, are meant for. This allows us to cut down on the interference effects generated by flooding the cells through the network and engaging nodes which need not be concerned in any manner with the transfer of an image which it will not use. Naive data dissemination in networks can lead to what is know as a broadcast storm problem, where the contention level of the network as a whole increases significantly and thereby reduces throughput of the complete network in general [6]. It is in this respect that we employ our REQ Suppression mechanism which reduces the number of transmissions for image cells percolating to the layers of nodes to the sink. Experiments conducted in [7] refer to irregular contours with respect to packet propagation and link layer metrics. Deluge is a network reprogramming protocol which achieves bulk data dissemination in sensor networks. Deluge can only work with homogenous environments, and is supported by the latest versions of TinyOS. Deluge borrows ideas from protocols such as SPIN-RL [8], an epidemic based algorithm for broadcast networks that make use of a 3-phase handshake protocol. We co-agulate the handshake mechanism in SIMTorrent by suppressing REQ messages from nodes of similar type, to reduce network traffic over the gradient tree. Trickle is another protocol, from which Deluge borrows concepts [9]. We too borrow concepts from Deluge to support our reliable data transmission paradigm, however we also include concepts such as exponentially increasing windows for the receiver to send a single ACK for a number of cells received. This concept is similar to TCP windowing mechanisms. Protocols such as Pump Slowly Fetch Quickly (PFSQ) [10] and Reliable Multi Segment Transport [11] are selective NACK based approaches for transport protocols designed for wireless sensor networks. As the cost of end-to-end recovery is extremely high, all these protocols base themselves on a hop-by-hop data dissemination paradigm and use 16 bit CRCs for error checking at each transmission. We too adhere to the hop-by-hop paradigm, however we manage to cut down on the number of ACKs by our exponential windowing scheme. TinyOS [12], includes support for a XNP [13] which provides for only a single hop solution to network reprogramming needs of the user. It necessitates that all nodes be within a bidirectional range of the source. MultiHop Over the Air Programming (MOAP) presents a more comprehensive approach to network programming providing multihop support for network re-tasking of sensors [14]. SIMTorrent is more complex than MOAP or Deluge for that matter, as it fragments images meant for different types of nodes into different cell sizes, unlike what is done in Deluge. The wireless link over which the cells are going to be transmitted control the cell size of the transmissions that are going to occur over that particular channel. MOAP and Deluge do not include these concepts.

III. SIMTorrent

SIMTorrent is a hybrid Push-Pull based protocol, it operates by an initial message flood to all nodes in the network in order to inform them of a new image arriving at the sink. This information is passed in the form of a Image Vector (IV), containing concise data about, the image version number and the type of node for which it is meant. This simplistic mechanism provides us with a low overhead methodology for aggregating reverse interest propagation by the nodes that become interested in the image that as just arrived. SIMTorrent handles heterogeneous collections of sensor nodes in a network using a plethora of mechanisms. Whenever a new image arrives at the sink in the network, an IV is propagated through the deployment area in order to inform all the nodes of this new piece of data. Nodes now ascertain if the image type is the same as their platform type, if it is they discover on the fly the version of the image that they are running in order to decide whether to ask for that image or not. Nodes which now decide to ask the image now broadcast REQ messages to their neighbors in their respective transmission ranges. The REQ Suppression mechanism makes sure that no more than one message is transmitted through to the corresponding levels of sensors. This is a basic aggregation strategy in order to cut down on the number of REQ messages for the same image percolating through the network. We address one of our main goals reliable and energy efficient dissemination of program images over the entire sensor network by using exponential window like schemes for ACK responses from the receiving nodes. This cuts down on nodes responding to each and every cell reception. Furthermore we use a common datum based scheduling policy to serve one or more nodes asking for pages of the same image. We attempt to first bring all the nodes to a common level that is by servicing the node which asks for the lowest numbered pages, first, until it asks for pages common to most other nodes. This allows us to use one local broadcast to disseminate the cell for a particular image in one burst instead of multiple transmissions, as in other network reprogramming protocols.

A. Data Representation

To manage larger than RAM page sizes we dynamically fragment and aggregate cells to form new bigger or smaller cell sizes. The factors governing this policy range from the RAM size of the target node, to the wireless data rate of the link between the sender and the receiver. Of course, the

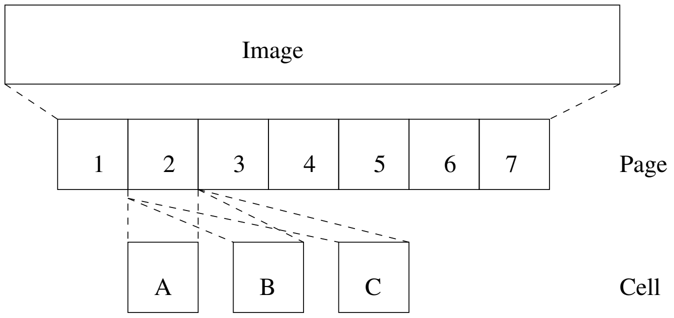

Fig. 1. Hierarchical segmentation of data image

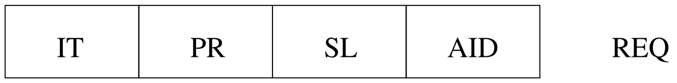

Fig. 2. The REQ message. IT: specifies the image type, 2 bits are sufficient for this field, since its purpose is to simply differentiate the type of node characteristics it has vis--vis the other nodes. PR: specifies page range for the particular image that it wants, 4 bits are reserved for this field. SL: this specifies the whether the node was in Hibernate state, following some sleep schedule prior to waking up. This bit allows us to inform the node receiving the REQ to first search locally for the pages in the PR, since they could have been passed onto it, or its neighbors while the sleep cycle was in effect. AID: specifies the aggregator node which should employ REQ aggregation, in an explicit manner.

image is divided up in logical partitions of fixed sizes called a page; the page however may be broken up into cells for actual transmission. A pictorial representation of this structure is presented in Figure 1.

The total image is divided up in equi-sized blocks [3], [4] called pages; the pages are sub divided into cells which are actually transmitted over the radio links. We have $S_{page} = NS_{cell}$ where S_{page} defines the page size of the image being disseminated and S_{cell} defines cell size used for transmission over a particular link. Cell size is determined by the link over which the cell has to be transmitted. Initial interest propagation is carried out with the IV. Once interest has been identified by nodes of requisite type, they reply back using the REQ message, see Figure 2, this message contains the page range the node is asking for and the image type it is asking for. We assume for simplicity that any particular point in time no attempts are being made to program sensors with multiple versions of the same image. Obviously, it is quite logical to assume that the network designer will not try to flood multiple versions serially of the same image as there is absolutely no benefit from doing so. Sensors do not evolve; they get programmed in a one-shot effort.

Once the REQ messages reach the REQ suppressor/aggregator node these messages are coagulated on the basis of their PRs and their ITs and passed further through the gradient tree formed. Once these REQ messages reach the sink, image propagation is triggered off in full swing. Image cells are passed onto the nodes which lie on the gradient tree, which is identified by reading off the aggregated AID fields from the REQ messages. The sink thereby has complete information regarding the last suppressing/aggregating node which has to receive the packet and hence unicast transmissions of the needed Image cells is initiated.

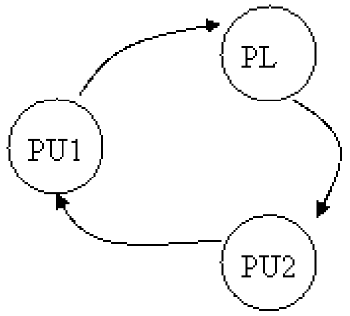

Fig. 3. The FSM modeling of the SIMTorrent protocol is simplistic and inherently deadlock free. PU1: this specifies the state corresponding to the PUSH phase 1, wherein the sink floods the IV. PL: this specifies the corresponding state to the PULL phase wherein the gradient tree is generated after stripping of the aggregated list of AIDs from the REQ messages at the sink. PU2: specifies the PUSH phase 2 state wherein the actual data in the form of cells is propagated through the network.

B. *The Protocol*

SIMTorrent is based on a hybrid push-pull based mechanism. At the very beginning as a new image is supplied to the sink it floods an IV throughout the entire network, This is the first phase push operation wherein the IVs leave the sink and inform all the nodes via a hop-by-hop propagation of the type of image available at the sink. This is followed by a Pull operation, the REQ messages are targeted back at the nodes which initially informed the interested nodes about the presence of the new image at the sink. The REQ cell defines the detailed description of the needs of the node, i.e. PR and IT, SL etc. Once the REQ messages have been aggregated and have been passed on to the sink, the sink determines which cells of the image it needs to disseminate in the network and does so. The gradient tree is actually a reverse interest propagation for the nodes which identify he image as being useful for itself or are acting as aggregator/suppressor nodes. There are therefore three basic stages for FSM modeling of the entire protocol, as presented in Figure 3. The basic ideas used in SIMTorrent are further depicted in Figure 4.

The nodes in the sensor network need to maintain some local routing information about their surroundings and this is achieved in SIMTorrent via exchange of 1-hop neighbor info among neighboring nodes which leads to a 2-hop neighborhood vision at each node and thereby provides a low-overhead mechanism for expanding the extremely limited 1-hop routing topology view which is of little use. Once all nodes have their 2-hop neighborhood information they can participate in the suppression/aggregation processes in two distinct manners.

Suppression: Nodes that are interest in the IV that they receive transmit REQ messages to their neighbors from whom they received the IV, via a local broadcast. Multiple nodes in turn receiving these REQ messages wait for a random backoff period in order to attempt to piggyback REQ messages from

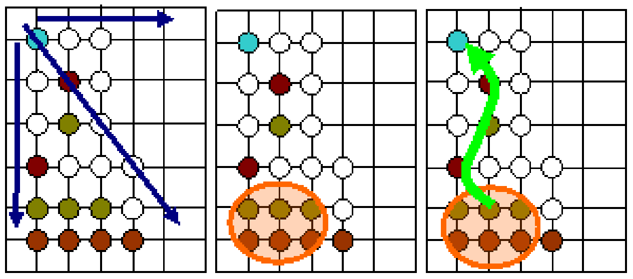

Fig. 4. 4A: depicts the PU1 state as depicted in the previous figure. In this state the IV is propagated throughout the network. 4B depicts the identification of neighbors of same type who are interested in the image and would like to send REQ messages. 4C depicts the gradients tree via which aggregation/suppression of REQ messages can occur and can finally reach the sink.

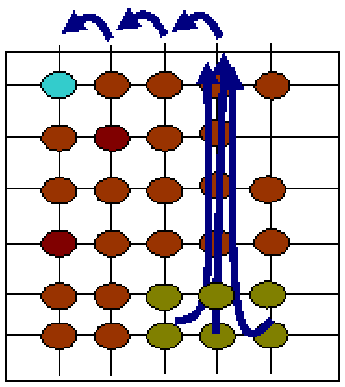

Fig. 5. Represents the aggregation/suppression of REQ messages from the ”interested” nodes towards the sink.

nodes which still might be processing the IV. The first node whose timer expires is aggregates the REQ messages it has heard so far and passes them on to the nodes from which it heard the IV in the same recursive manner.

Aggregation: Nodes interested in the IV, having already exchanged 1-hop neighbor tables and having a 2-hop view of the neighborhood, now designate a specific node from which they all heard the IV. This selection can be done on the basis of parameters like link quality, or low downtime values. An explicit AID is marked in the REQ message to let the chosen aggregator node know about its special purpose. All interested nodes unicast REQ messages to the aggregator henceforth. Note that since nodes now have 2-hop views, a maximum of 3 nodes (in a grid layout) in the immediately upper/closer to the sink level can be chosen by individual nodes. We explain this in detail in the next section. This is depicted well in Figure 5.

Another major feature of SIMTorrent is its capability to delegate authority to each pair of communicating nodes to decide optimum cell size. Nodes communicating with each other can easily ascertain the capabilities of the link among them by reading off the IT field in the REQ message. We assume that the parameters that correspond to each type of sensor e.g. RAM size, Flash memory capacity, Radio Channel data rate are stored inside the nodes, thereby all the

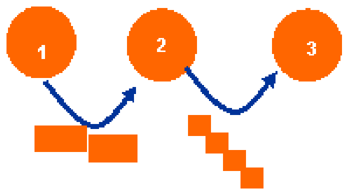

Fig. 6. Represents cells of different sizes being transmitted over links between different types of nodes with varied capabilities. This is essential in order to optimize the RAM size, Flash memory size, computational capabilities of different types of sensors.

suppressor/aggregator node needs to do is read off the IT filed in the REQ message to know which kind of node it’ll be passing the cell to and thereby dynamically fragment incoming cells to create smaller cell sizes or coagulate incoming cells to form larger chunks to reduce transmissions. This methodology has not been employed by any other network reprogramming protocol that we know of. This is represented in Figure 6.

SIMTorrent also employs a common datum scheduling policy explicitly meant to cut down on the number of transmissions while servicing multiple nodes from a common source. Existing network reprogramming protocols like Deluge [3], [4] use a round-robin policy. We employ a common datum based approach, in which we first service the node that asks for the lowest numbered cell, and keep servicing it until its cell number being requested is the same as that of the cell numbers being requested by the other members interested in that image. Once we reach this state we can use one local broadcast to service more than one node in one shot. This reduces the number of effective transmissions for the same cell that needs to reach a number of nodes interested in the image. Also, one of the novel features of SIMTorrent is to handle sleep cycles and allow for dissemination of cells after the quorum of nodes that was in hibernate state comes back on. This is achieved by asking all the nodes which want to follow a sleep schedule to select a leader among them, again based on the 2-hop neighborhood information that the nodes possess due to exchange of 1-hop local information. Now, each of these chosen nodes are responsible for caching pages in its local/on-board memory for the nodes that are sleeping. This is a rudimentary approach, which we intend to improve upon, however it is still sufficient to allow for nodes to sleep and yet on waking up to probe a known node for cells that might be in its cache.

IV. Experimental Results

In this section, we will present the simulation results we gathered from the SIMTorrent simulator [15]. It is a simplified, home-grown and discrete simulator written in C++. Although it does not have RTS/CTS mechanism built-in, it does implement a simple error model based on the number of neighbors each node has, ie. the more neighbors a node has, the more likely the transmitted packets to its neighbor nodes will be dropped due to bandwidth contention.

All the simulations were set to run for 30 minutes. We

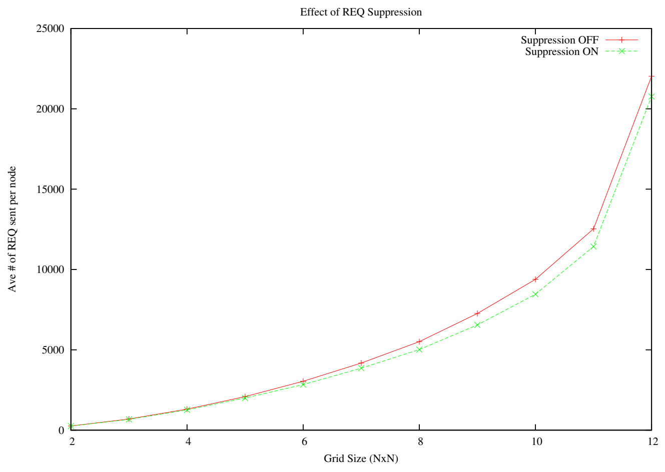

Fig. 7. REQ Suppression

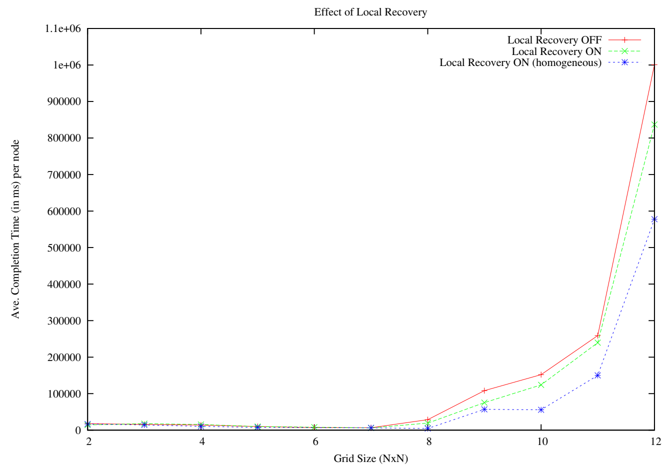

Fig. 8. Local Recovery

simulated only 3 types of node, each with image size of 2000 bytes. Each node type was assigned randomly at run-time. Each simulation had different grid size ranged from 2x2 up to 12x12 with different combination of performance optimizations. In each simulation, all nodes reliably received a completed image based on their node types/platforms.

We first show that SIMTorrent's performance based on the effect of REQ Suppression, Local Recovery, On-demand Packet Caching and Push Model individually. And then, we combine all optimizations together and show that it performs better than any individual one.

A. *REQ Suppression*

As mentioned in Section III, when each nodes receive a REQ message, it randomly backs off and listens to its neighbor to see if the same REQ is being transmitted. If it does, the node simply ignore the REQ message, otherwise, it would go ahead and broadcast the message to its neighbors. This mechanism would avoid transmitting relundant REQ messages overheard by its neighbors.

Figure 7 shows the average number of REQ sent by each node with and without REQ Suppression in various topology. At first glance, the differences seem insignificant, however, if each node can suppress small amount of REQs, overall, the total saving of the number of REQs in the networks can be huge. For example, in the case of 12x12, each node with REQ Suppression mechanism sent 1267 less messages on average comparing to those without. In other words, the sink node would have received 1267 * (12 * 12 - 1) [1], which is about 182K, more REQs if there were no REQ Suppression mechanism in place.

B. *Local Recovery*

All REQs are broadcasted and eventually the REQs go all the way and reach the sink node which in turn replies the REQs by unicasting DAT packets to the requested nodes. However, it is extremely inefficient to do so. If a REQ reach some intermediate nodes that happen to have the requested pages and cells, they could serve the REQ immediately instead of further propagating the REQ to the sink. Figure 8 shows that Local Recovery indeed shortens the average completion time. In the case of 12x12, nodes with Local Recovery enabled finishes about 2 minutes earlier on average for each node.

The reason that prevents SIMTorrent to take full advantage of Local Recovery is because of nodes' heterogeneity. In other words, a node might be surrounded by other types of nodes, in which case, Local Recovery is completely useless. To further confirm this, we also plotted a 3rd line (in blue) which showed that the average completion time is significantly shorter in a homogenerous sensor networks.

C. *On-demand Packet Caching*

In order to avoid forwarding duplicated packets (including REQs and DATs), SIMTorrent implements Packet Caching on each node. However, since sensor nodes are typically small and thus resource constrained, they usually equipped with very limited amount of RAM, so, how to utilize the cache in an efficient manner is extremely important to SIMTorrent. In our simulation, we use 1 page size (ie. 1104 bytes) [2] as the cache size for each node.

In SIMTorrent, we first implement FIFO caching replacement policy. ie. when the cache becomes full, the first one in the cache is replaced by the new one regardless the type of the packet. We also implement On-demand Packet Caching which always tries to replace the one with the same type. Intuitively, this should be very helpful if a node is surrounded by one type of neighbors more than others. However, Figure 9 shows it otherwise.

D. *Push Model*

As mentioned in Section III, SIMTorrent is a hybrid Push-Pull based protocol. Not only it pushes the IV (Image Vector)

[1]We need to exclude the sink node itself.

[2]Like Deluge, each page is divided by 48 cells, each cells has a data payload of 23 bytes.

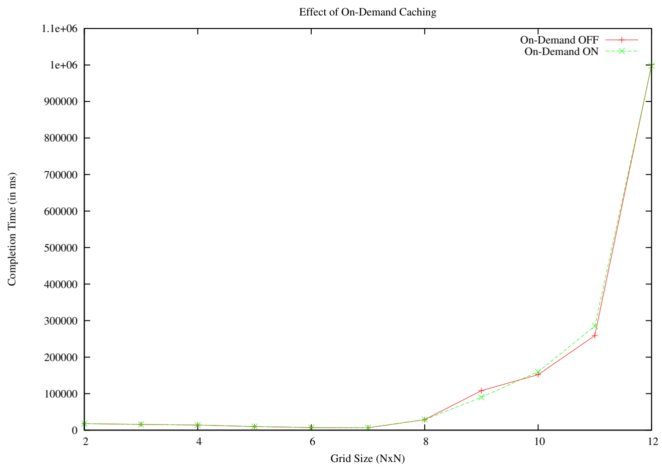

Fig. 9. On-demand Packet Caching

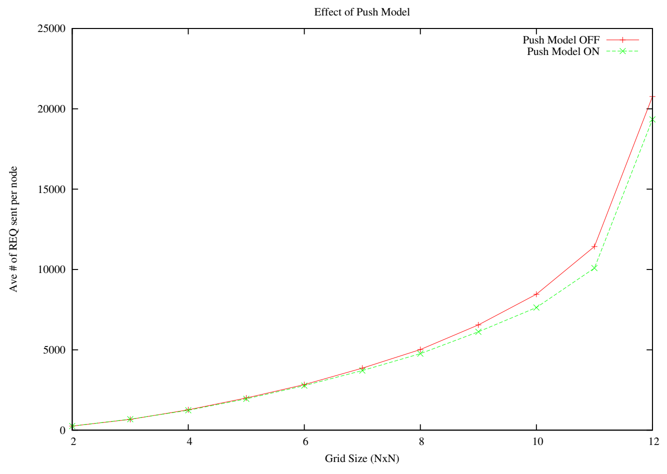

Fig. 11. Push Model (REQ)

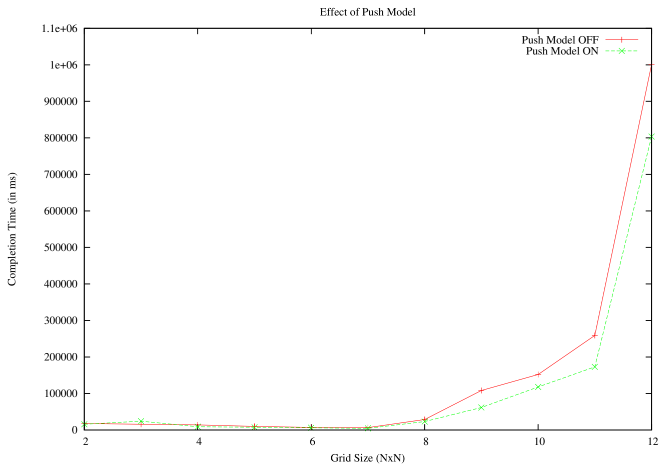

Fig. 10. Push Model

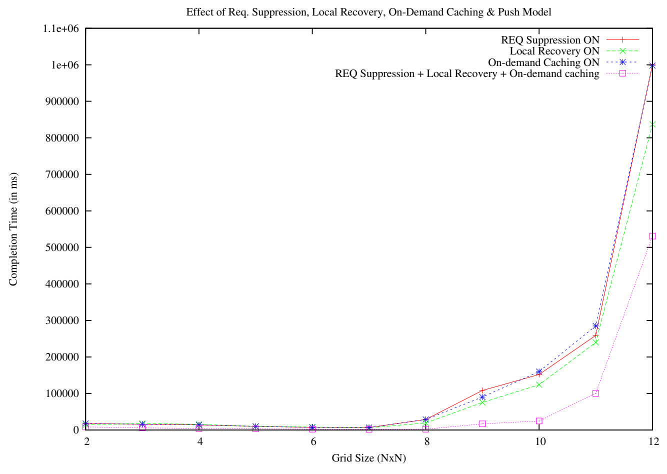

Fig. 12. REQ Suppression + Local Recovery + On-demand Caching + Push Model

to all the nodes, then each nodes decide individually which image it would like to download, it also tries to push more DAT packets to its requesting neighbors in order to achieve higher throughput. Figure 10 shows that the average completion time per node is shorter when SIMTorrent tries to push **one** more DAT packet whenever available. The figure also shows that the Push Model optimization only kicks in after 8x8.

Figure 11, on the other hand, shows that the number of REQs sent per each node with Push Model enabled is less than those that are not. Again, this figure shows the average number of REQs sent by each node.

E. Put them all together

In Previous subsections, in order to see how significant each optimization is, we showed those individually. Figure 12, on the other hand, shows all optimizations into action together which improves the performance even more.

V. Future Work

As a natural extension of our research effort, we would attempt to include other better caching strategies in SIMTorrent. This would allow for nodes which would wake up from ”sleep cycles” to probe only a specific set of neighbors instead of making a local broadcast to get cells which might have been distributed while they were asleep. We intend to analyze SIMTorrent over a much larger topology, and attempt to break out of the grid model to a more generic non-symmetric graph based model. Also, we would like to determine how many DAT packets should be pushed in order to achieve better throughput without increasing too much bandwidth contention. Furthermore we will attempt to investigate scenarios using simulations consisting of multiple sinks pumping in images at the same time into the network.

VI. Conclusion

Our protocol, SIMTorrent is a hybrid Push-Pull based design which involves a plethora of techniques to reduce the number of effective transmissions that are needed for the dissemination of program images over a sensor network. SIMTorrent can handle heterogeneous node deployment scenarios, and features REQ suppression/aggregation, common datum scheduling and TCP like exponential window based ACK scheme. All these features allow for SIMTorrent to save on the number of transmissions as proved by our simulation results. We show conclusively the benefits achieved by using SIMTorrent over a more generic protocol which does not use these enhancements.

REFERENCES

[1] Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., and Cayirci, E, "A Survey on Sensor Networks, IEEE Communications Magazine," Vol. 40, No. 8, pp. 102-116, August 2002.

[2] D. Estrin, D. Culler, K. Pister, G. Sukhatme. "Connecting the Physical World with Pervasive Networks", IEEE Pervasive Computing archive, Vol. 1, No. 1, pp. 59-69, January 2002.

[3] S. Neema, A. Mitra, A. Banerjee , W. Najjar, D. Zeinalipour-Yazti, D.Gunopulos, V. Kalogeraki. "NODES: A NOVEL SYSTEM DESIGN FOR EMBEDDED SENSOR SYSTEMS", SPOTS Track, IPSN 2005.

[4] A. Talukder, R. Bhatt, L. Chandramouli, T. Sheikh, R. Pidva, and S. Monacos, "Autonomous Resource Management and Control Algorithms for Distributed Wireless Sensor Networks", To appear in The 3rd ACS/IEEE International Conference on Computer Systems and Applications - January 2005.

[5] Jonathan W. Hui and David Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale", The 2nd ACM Conference on Embedded Networked Sensor Systems (SenSys'04), November 3-5, 2004.

[6] S., Y. Ni, Y. C. Tseng, Y. S. C and J. P. Sheu, "The broadcast storm problem in a mobile ad hoc network", Proc. Of 5th Annual ACM/IEEE International Conf. on Moble Computing and Networking, PP 151-162, ACM Press 1999.

[7] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin and S. Wicker, "Complex behavior at scale: An experimental study of low power wireless sensor networks". Technical Report UCLA/CSD TR02-0013, UCLA 2002.

[8] J. Kulik, W. R. Heinzelman, and H. Balakrishnan. "Negotiation-based protocols for disseminating information in wireless sensor networks". Wireless Networks, 8(2-3):169-185, 2002.

[9] P. Levis, N. Patel, S. Shenker, and D. Culler. "Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks". Technical report, University of California at Berkeley, 2004.

[10] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. "PSFQ: A reliable transport protocol for wireless sensor networks". In Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, pages 1-11. ACM Press, 2002.

[11] F. Stann and J. Heidemann. "RMST: Reliable data transport in sensor networks". In Proceedings of the First International Workshop on Sensor Net Protocols and Applications, pages 102-112, Anchorage, Alaska, USA, April 2003. IEEE.

[12] University of California, Berkeley. Tinyos. http://www.tinyos.net/, 2004.

[13] J. Jeong, S. Kim, and A. Broad. "Network Reprogramming". University of California at Berkeley, Berkeley, CA, USA, August 2003.

[14] T. Stathopoulos, J. Heidemann, and D. Estrin. "A remote code update mechanism for wireless sensor networks". Technical report, UCLA, Los Angeles, CA, USA, 2003.

[15] SIMTorrent, http://www.cs.ucr.edu/~weesan/sensor_networks/simtorrent-0.0.1.tar.gz, 2005.