An Implementation of a Data-transmission Pipelining Algorithm on Imote2 Platforms

Xu Li[a], Siavash Dorvash[b], Liang Cheng[a], and Shamim Pakzad[b]

[a]Department of Computer Science and Engineering, Lehigh University
[b]Department of Civil and Environmental Engineering, Lehigh University

ABSTRACT

Over the past several years, wireless network systems and sensing technologies have been developed significantly. This has resulted in the broad application of wireless sensor networks (WSNs) in many engineering fields and in particular structural health monitoring (SHM). The movement of traditional SHM toward the new generation of SHM, which utilizes WSNs, relies on the advantages of this new approach such as relatively low costs, ease of implementation and the capability of onboard data processing and management. In the particular case of long span bridge monitoring, a WSN should be capable of transmitting commands and measurement data over long network geometry in a reliable manner. While using single-hop data transmission in such geometry requires a long radio range and consequently a high level of power supply, multi-hop communication may offer an effective and reliable way for data transmissions across the network. Using a multi-hop communication protocol, the network relays data from a remote node to the base station via intermediary nodes. We have proposed a data-transmission pipelining algorithm to enable an effective use of the available bandwidth and minimize the energy consumption and the delay performance by the multi-hop communication protocol. This paper focuses on the implementation aspect of the pipelining algorithm on Imote2 platforms for SHM applications, describes its interaction with underlying routing protocols, and presents the solutions to various implementation issues of the proposed pipelining algorithm. Finally, the performance of the algorithm is evaluated based on the results of an experimental implementation.

Keywords: Data-transmission, Pipelining algorithm, Multi-hop communication, Wireless sensor networks.

1. INTRODUCTION

A review on the literature of the Wireless Sensor Network (WSN) applications in the civil engineering shows the increasing interest of researchers in the use of WSN in Structural Health Monitoring (SHM) of bridge infrastructure. Part of this interest is because of the convenience in the deployment of wireless units for sensing and communication and part is due to the crucial health condition of existing bridge structures [1]. Although WSN serves an efficient, low-cost and easy to install tool for health monitoring of infrastructure, there are some drawbacks that limit their broad application in this SHM. Power maintenance and effective communication are two major challenges in the current status of WSN. To overcome these difficulties in the application of WSN, many studies have focused on these topics in recent years. Developing effective power usage strategies ([2] and [3]), energy harvesting approach ([4] and [5]), transmission of power through radio signals ([6]) are few examples of researches on the power maintenance topic.

While many researchers have concentrated on the limited power in WSN, less effort has been observed in addressing the communication efficiency issues found in on-site WSNs deployed at bridges. In the particular case of long span bridge monitoring, when WSN with single-hop data communication is applied, the radio range can considerably limit the total length of the network. As an example of this limitation, in one of the recent large deployments of WSN for health monitoring of a cable-stayed bridge [7], restriction of radio range in single-hop scheme enforced dividing the network

Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems 2011, edited by Masayoshi Tomizuka, Chung-Bang Yun, Victor Giurgiutiu, Jerome P. Lynch, Proc. of SPIE Vol. 7981, 798104
© 2011 SPIE · CCC code: 0277-786X/11/$18 · doi: 10.1117/12.880451

into two sub-networks with separated base stations. In such deployment, synchronization of the sub-networks becomes impossible which could affect the final obtained results. An approach addressing this limitation is multi-hop data communication. In this transmission scheme, two nodes which are not in the direct radio range, communicate using nodes in-between [8]. Example protocols developed for multihop routing include Beacon Vector Routing (BVR) [9], Virtual ring routing (VPR) [10], MintRout [11], General Purpose Multi-hop (GPMH) and Single-Sinked Multihop (SSMH) [12].

Further enhancement of communication in long linear networks is the reuse of the bandwidth by applying pipelining approach in which several nodes transmit data at the same time. The basic idea of pipelining is to schedule the behavior of the nodes along the multi-hop communication path such that controlling the sending rate to maximize the channel usage and minimize the energy consumption, delay performance and packets collisions [13]. The important advantage of pipelining, over the simple multi-hop approach, is its impact in the network agility. [14] shows the significance of network's agility in structural monitoring application.

Despite the substantial improvement that pipelining brings to the performance of wireless networks, literature shows little attention to its application in long span bridge monitoring. Deployment of a network of 64 wireless sensors on the Golden Gate Bridge [8] is the only significant work which has incorporated pipelining for transmitting data across the bridge. The hardware platform of that work was Micaz which had been one of the promising platforms at that time. Since both hardware and software platforms of wireless sensors evolve rapidly, it is necessary to update and enhance the protocols in this research field. Presented in [13] is a pipelining algorithm that gives consideration to both channel usage improvement and deployment simplicity, which is called "Pushpin" here. This paper presents the implementation of Pushpin on Imote2 platform which is a popular hardware platform in the field of SHM. The pipelining algorithm is integrated into a scheduling component running on the Imote2 nodes. Through a series of calibration test, a test-bed is provided to compare the performances between the Pushpin pipelining scheme and a prevalent reliable transmission protocol used in WSN field deployment for SHM [15]. The results show that the pipelining scheme gains a significant edge in session throughput as the WSN route includes multiple hops.

2. THEORY OF THE ALGORITHM

Pipelining is a scheduling strategy that greedily utilizes the wireless channel resource. Once a sender assumes that the wireless channel is idle, it will keep pumping packets to the network without stopping to confirm the successful reception of each packet. The pipelining algorithm details can be depicted in the pseudo code below.

```
Variables: p, q,   //pointers to packets

t;          // reading of timerwhile(true)
{
        if (!getPacket(p))
        goto CLK;
        p -> receiverId = RECEIVER;
        p -> packetVersion = ver++;
        p -> senderId = TOS_LOCAL_ADDRESS;
BC:             broadCast(p);
CLK:    for (t=T_WAIT; t>=0; t--)
        {
        if (receivePacket(q) && q->packetVersion >= ver[q->senderId])
                {p=q;
                goto BC;}
        }
}
```

The behavior of this pseudo code is explained in the flow chart Figure 1. At the beginning of every iteration, the program packs up a packet from the data waiting to be sent. The flags and variables in the packet structure are set according to the source and destination nodes id. Then the packet is sent out by broadcasting, and at the same time a timer is set to count down from a constant T_WAIT. During this period, if there is another packet arriving from the wireless channel, the program will first check whether the packet has been sent from this node before. If not, the packet will be re-broadcasted and the clock will be reset to T_WAIT. After the timer signals a timeout, the program will fetch another packet from the local data pool.

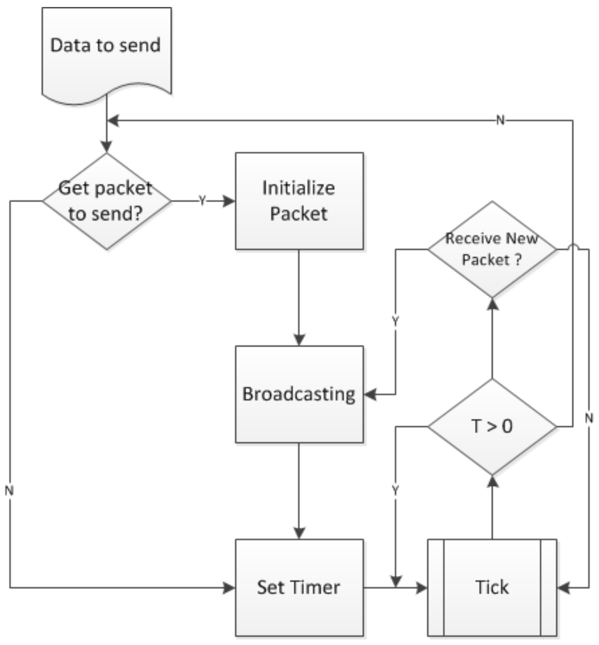

Figure 1 Flow Chart of Pushpin Mechanism

The waiting period T_WAIT is a crucial parameter in this algorithm, which stands for the time it takes for a packet going out of the interference range of the sender (as shown in Figure 2). This value is related to multiple factors including the transmitting power of each node, network density and the processing speed. In a real world implementation, a way to estimate this value is multiplying the one hop delay by some empirical constant.

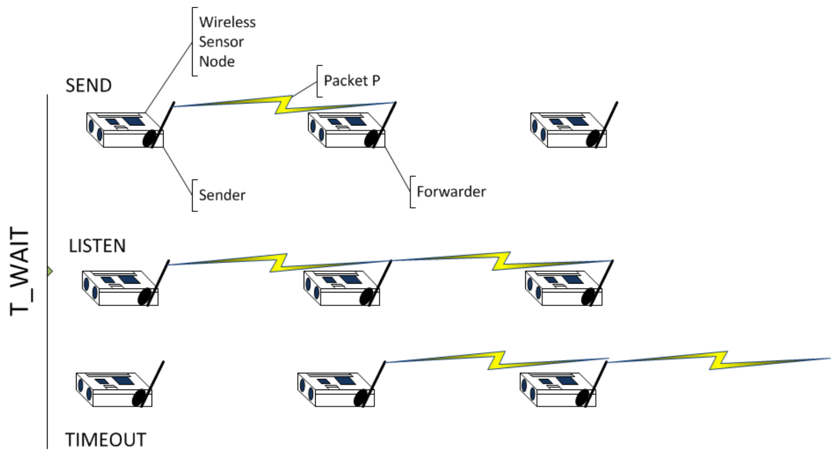

Figure 2 Example of an Operation Period by the Pipelining Algorithm

3. IMPLEMENTATION

We name our implementation of the algorithm on Imote2 platform “Pushpin”. Our goal is to provide a TinyOS component which is comparable to the ReliableComm in current release [15] , plus the function of multi-hop pipelining. The interface we provide is similar to the one from the ReliableComm, therefore it will be convenience for current user of ReliableComm to switch to our application. Some stats of both components are listed in the table below.

Table 1. Parameter Setting Comparison

	Pushpin	ReliableComm
Data Packet Size	29 Bytes	127 Bytes
Payload Size	18 Bytes	96 Bytes
Range	Multi hop	Single hop
Require ACK	No	Yes

The reason we choose a relatively small packet size for Pushpin is because of the compatibility issue. The ReliableComm and the entire SHM tools family is built upon TinyOS 1.x system, which has a data packet structure TOS_MSG with a maximal packet length 127 Bytes. While in TinyOS 2.x, this data structure evolves to an abstract data types (ADT), with platform specific header and footer, leaving a data section with a default length of 29 Bytes. We restrict the packet size so that the Pushpin application can be easily upgraded to TinyOS 2.x version if needed. This setting may hinder the transmission speed with extra processing work load. In other words, the throughputs we gain here can be treated as a lower bound. Since Pushpin offers an interface similar to ReliableComm.send, it takes four input parameters: recipient list, recipient number, data pointer and data length. Despite the similarity in the interfaces the two components handle the parameters differently: the ReliableComm unicasts the packets to the neighbors that are in the recipient list while the Pushpin broadcasts the packets at the fastest available speed in a multi-hop manner. To further reduce the time consumption for network agility, Pushpin does not require acknowledgement for each packet. (In future release we may include features like NACK, i.e. Negative ACK, that indicating the potential transmission failure to handle the packet loss.)

4. EXPERIMENTAL EVALUATION

We write our own application TestPushpinM to test the efficiency of the pipelining. This application runs a simple data transmission tasks: taking a local data buffer, sending it through ReliableComm.send. At the receiver end, ReliableComm.receive also takes charge of receiving the packets and reconstructing the data. If there are other receivers beside the local node, the receive event will also trigger the forwarding process. In either case the successful rate of packet transmission will be measured and displayed.

4.1 Calibration

The default transmission range for Imote2 is around 30 meters. To have the multi-hop test in a manageable scale, we lower the transmission power of the transceiver CC2420 from 0 to -24dBm. To determine the new communication range under this setting, we construct a calibration test. In this test, two Imote2 nodes are separated by a certain distance, one of them sends 100 packets to the other one, and the receiver reports the packet number it received. We repeat 10 times for each distance, and the result is plotted in Figure 3. From the graph we can identify that the link state has three stages when the distance increases: first one is from 15 to 35 cm, the successful rate drop from perfect to around 0.75; the second one is from 35 to 75cm, where the reception success rate jitter around 0.75; and the third one is further than 75cm, where the success rate drop dramatically to 0. To get an acceptable packet loss rate, we choose 15cm as the span between two neighbor nodes.

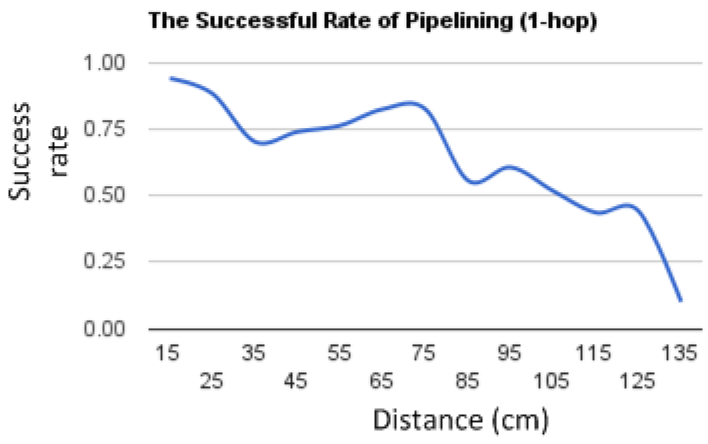

Figure 3 One Hop Link Quality

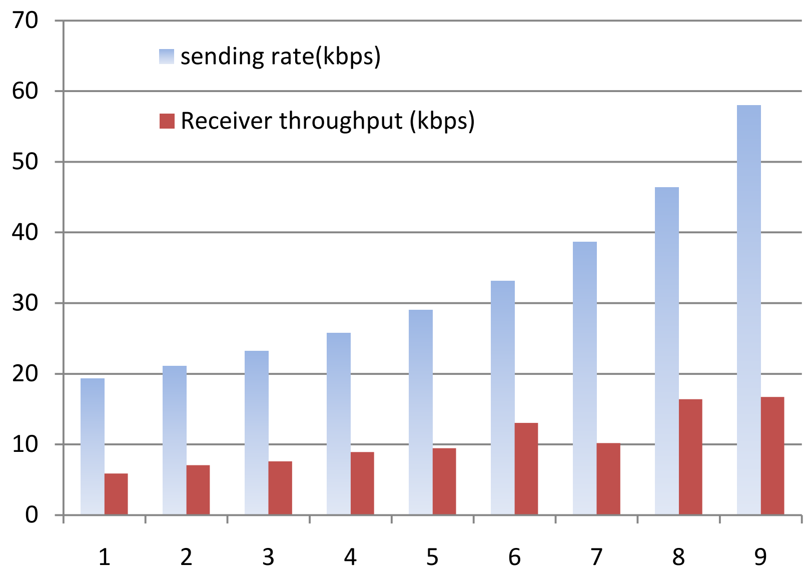

Figure 4. Throughputs Observed at Both Sender and Receiver Sides

4.2 Saturation Test

Another crucial parameter we need to decide for the Pushpin application is the sending rate. Besides the interference constraint, the wireless communication among wireless nodes is also bounded by the processing speed of the microprocessors. That is, when the sending speed exceed some threshold, the throughput observed at the receiver side will not keep increasing since the processor is occupied in handling current packets rather than new arriving ones. This phenomenon is depicted in Figure 4, where the gray bars stand for the sending data rate (throughput from sender side), and the solid bars stands for the corresponding throughput observed at the receiver side. From the figure we can see that the receiving throughput does not increase significantly after the sending speed reach 30kbps. Thus we choose 30kbps as the default sending speed for Pushpin. One may notice that this value is only a portion of the maximum transmission capacity of cc2420, which is 250kbps. The reason for this relatively low throughput is two-fold: firstly, as mentioned above, we choose the lowest transmitting power for CC2420; secondly, we only use a small portion of the TOS_MSG load capacity (29/127) in this application.

4.3 Comparison Test

The comparison between the ReliableComm and the Pushpin is conducted upon a small multi-hop network. Since the ReliableComm treat large and small size of data differently (using ReliableCommS and ReliableCommL correspondingly), in order to keep the complexity of data processing from affecting the throughput measurement, we choose a small data size (100 bytes) for ReliableComm to transmission. Pushpin does not have such restriction on data size, so the test data is set to a larger chunk (1600 bytes) to rule out of randomness. The result is shown in Figure 5. As we iterated before, the throughput values listed here are the saturated throughput under a power constraint and software cost. Thus we define it as "saturated effective throughput" From the result we can see a clear pattern of the throughput: in the single-hop scenario, the ReliableComm performs better than Pushpin, while as the hops number get larger, the burden of ACK exchanges greatly hinder the data transmission of ReliableComm, which lead to a substantial performance lag when compared with Pushpin.

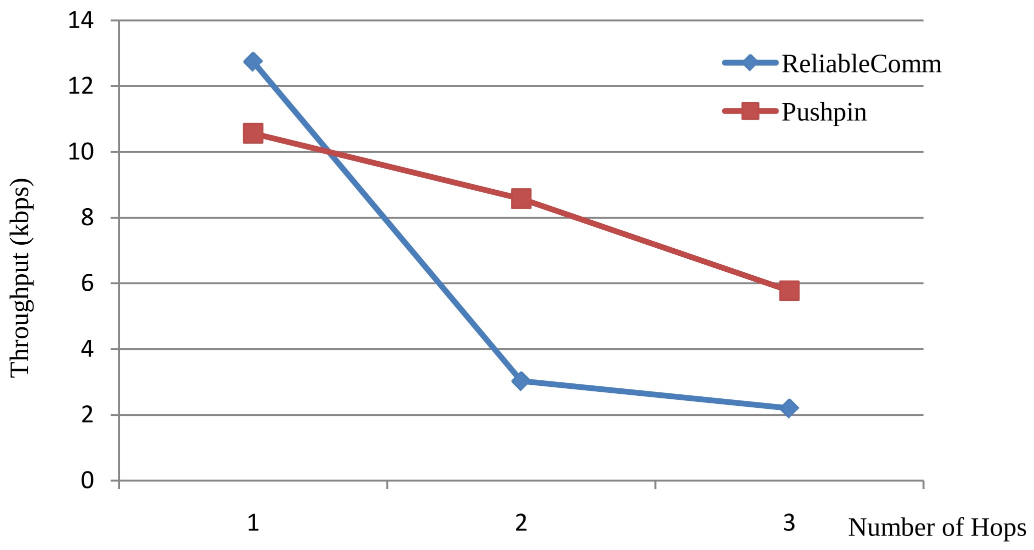

Figure 5. Saturated EffectiveThroughput in Multi-hop Cases

5. CONCLUSIONS

In this paper we present a design of pipelining data transmission algorithm and its implementation on Imote2 wireless sensor network platform. Through comparison with a reliable transmission component, we prove that as a scheduling strategy, pipelining algorithm out performs traditional one-to-one reliable transmission algorithm in multi-hop transmission throughput. The greedy algorithm of the speed control, which is the core of pipeling, turns out to be a very effective method for data collection. In the future, we will extend this Pushpin component to accommodate some real sensing application like RemoteSensing. Some new features will be brought in. For example, an NACK mechanism is expected to be added in to handle packet loss. Also an automatic link state estimator is needed for the dynamic calculation of sending speed.

6. ACKNOWLEDGEMENT

The research described in this paper is supported by the National Science Foundation through Grant No. CMMI-0926898 by Sensors and Sensing Systems program. The authors thank Dr. Shih-Chi Liu for his support and encouragement.

REFERENCES

[1] U.S. department of transportation, "Status of the Nation's Highways, Bridges and Transit: Conditions & Performance," Report to Congress, United State (2008).

[2] Lynch, J. P., Sundararajan, A., Law, K. H., Carryer, E., Farrar, C. R., Sohn, H., Allen, D. W., Nadler, B., and Wait, J. R., "Design and Performance Validation of a Wireless Sensing Unit for Structural Health Monitoring Applications," Structural Engineering and Mechanics, Vol. 17, No. 3, 393–408 (2004).

[3] Gao, Y., Spencer, Jr., B.F., and Ruiz-Sandoval, M. "Distributed Computing Strategy for Structural Health Monitoring." Journal of Structural Control and Health Monitoring, 13(1), 488-507 (2006).

[4] Wang, M., Chang, P. C., and Newcomb, R., "Power Scavenging from Highway Bridge Vibration," Proc., International Conference on Structural Health Monitoring and Intelligent Infrastructure, Tokyo, Japan, Vol. 2, 1117–1126 (2003).

[5] Sodano, H. A., Inman, D. J., and Park, G., "A Review of Power Harvesting from Vibration using Piezoelectric Materials," Shock and Vibration Digest, Vol. 36, No. 3, 197–205 (2004).

[6] Wang, L. and Yuan, F.G. "Active damage localization technique based on energy propagation of Lamb waves," Smart Struct. Syst., 3(2), 201-217 (2007).

[7] Jang, S., Jo, H., Cho, S., Mechitov, K., Rice, J.A., Sim, S.H., Jung, H.J., Yun, C.B., Spencer, Jr., B.F. and Agha, G. "Structural health monitoring of a cable stayed bridge using smart sensor technology: deployment and evaluation," Smart Struct. Syst., 6(5-6), 439-459 (2010).

[8] Pakzad, S.N., Fenves, G.L., Kim, S., and Culler, D.E., "Design and Implementation of Scalable Wireless Sensor Network for Structural Monitoring," ASCE Journal of Infrastructure Engineering, 14(1):89-101 (2008).

[9] Fonseca, R., Ratnasamy, S., Zhao, J., Tien Ee, C., Culler, D., Shenker, S. and Stoica, I., "Beacon vector routing: scalable point-to-point routing in wireless sensornets," In NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation, pages 24–24, Berkeley, CA, USA, (2005).

[10] Caesar, M., Castro, M., Nightingale, E. B., O'Shea, G., and Rowstron, A., "Virtual ring routing: network routing inspired by dhts," In SIGCOMM '06: Proc. of the 2006 conf. on Applications, technologies, architectures, and protocols for computer communications, pages 351–362, New York, NY, USA, (2006).

[11] Woo, A., Tong, T., and Culler, D., "Taming the underlying challenges of reliable multihop routing in sensor networks," In Proc. of the first international conference on Embedded networked sensor systems, pages 14–27. ACM Press, (2003).

[12] Nagayama, T., Moinzadeh, P., Mechitov, K., Ushita, M., Makihata, N., Ieiri, M., Agha, G., Spencer Jr, B. F., Fujino, Y. and Seo, J. W., "Reliable multi-hop communication for structural health monitoring," Smart Structures and Systems, Vol. 6, No. 5-6, 481-504 (2010).

[13] Li, X., Dorvash, S., Cheng, L. and Pakzad, S. N., "Pipelining in Structural Health Monitoring Wireless Sensor Network," Proc., SPIE conference on Sensors and Smart Structures Technologies for Civil, Mechanical and Aerospace Systems, San Diego, CA, USA (2010).

[14] Pakzad, S. N., Cheng L., "Agility of Wireless Sensor Networks for Earthquake Monitoring of Bridges," Proc., 7th International Workshop on Structural Health Monitoring; From System Integration to Autonomous Systems, Lancaster, Pennsylvania 17602 U.S.A (2009).

[15] Software - Structural Health Monitoring at the University of Illinois (http://shm.cs.uiuc.edu/software.html)