Research Article

Control Theory-Based Load Balancing for Wireless Sensor Network

Sanqi Zhou1 and Yinchao Chen2

1 National Engineering Laboratory of Next Generation Internet Interconnection Devices, Beijing Jiaotong University, Beijing 100044, China
2 Department of Electrical Engineering, University of South Carolina, Columbia, SC 29208, USA

Correspondence should be addressed to Sanqi Zhou; 10111020@bjtu.edu.cn

Received 11 November 2013; Accepted 24 January 2014; Published 6 March 2014

1. Introduction

Wireless sensor networks (WSNs) have been widely deployed in many industrial and military environments to inspect, detect, monitor, and track the targets according to specific application scenarios. There are many research areas that have been explored such as service availability, energy efficiency, routing scalability, and reliability [1, 2]. In WSNs, communication efficiency and power consumption are always the main obstacles to improving service availability and energy efficiency. The power consumption of node and link (i.e., the module which is used to establish and maintain the wireless channel) is highly related to the corresponding traffic volume; thus, one method to solve the tradeoff between communication efficiency and power consumption is to distribute the traffic uniformly across the network. For this purpose, many load balancing approaches have been proposed to address this issue.

He et al. [3] proposed a real-time communication protocol named SPEED, in which a stateless nondeterministic geographic forwarding (SNGF) algorithm was employed. SPEED can support soft real-time communication based on feedback control and SNGF. The control overhead and transient congestion are also well handled. However, SNGF forwards each stream among random concurrent paths. Thus, if only very few paths with the same cost are available between each pair of nodes, there will be very few paths for SNGF to choose. Actually, two paths with the same cost can rarely be found between two nodes. Puccinelli and Haenggi [4] proposed a cost-based routing protocol named Arbutus with a built-in load balancing strategy. Arbutus allows each node to choose a parent of a lower depth node as the route to sink such that the routing loops can be avoided during the construction of routes and the control overhead can also be greatly reduced. However, it is possible that some neighbors of a node may choose the same parent of their lower depth nodes. Thus, this “parent” may become the hot spot such that the traffic will be much higher on the links connected with the hot spot while lower on other links. Dai and Han [5] proposed a node-centric approach to reduce hot spots to achieve load balance in WSNs. The algorithm grows a spanning tree and then iteratively selects the branch with the lowest load to graft onto the unassigned border node which generates the highest load. This algorithm can balance the network traffic flexibly;
Second, we design an algorithm for the controller to schedule packets. It forwards the packets on different links depending on the current link utilization (LU).

Meanwhile, we simply discuss how to design a routing protocol to be compatible with our approach to ensure that no packet will be forwarded in a loop (e.g., named data network (NDN) can be employed [13, 14]).

There are two main merits of our approach. (1) There is no knowledge of the static accurate traffic demands needed for using in the model because the traffic can be balanced according to the real-time output of the controller. (2) The hot spot can hardly be formed because the algorithm balances the traffic by following the theory from which the controller is designed. The control theory ensures the actual LU to follow the expected one. Meanwhile, we also perform simulations to verify the effectiveness and performance of our approach versus other solutions.

The rest of the paper is organized as follows. Section 2 describes the control theory-based model and then the algorithm. Section 3 describes the simulation and analyzes the results. Section 4 concludes the paper and describes the future works.

2. Methodology

In this section, we first present the model and interpret how it works. Then, we describe the algorithm which is designed based on the controller. Finally, the employed routing protocol is also described.

2.1. Theoretical Model. The model is established for each router. The main performance of load balancing is maximum link utilization (MLU); thus, we set the aimed link utilization as the reference input of the model while the current forwarded packet size in a time unit is set as the output. Meanwhile, the difference between the reference input and the output of feedback is used as the input of the controller. The output of the controller is the way to schedule packets in the router. The feedback element is responsible for converting the real-time packet size into LU. Notation section shows the notations used in this paper. Figure 1 shows the block diagram of the model based on control theory [12].

In notation section, each notation with variable $s$ is the Laplace transformation [15] of the corresponding time domain variable. We can easily get the model from Figure 1, and it can be stated as follows:

$$Y(s) = \left[ R_{LU}(s) - H(s)Y(s) \right] G_c(s) G_R(s),$$  \hspace{0.5cm} (1)

then, we derive the controller

$$G_c(s) = \left[ R_{LU}(s) - H(s)Y(s) \right] G_R(s),$$  \hspace{0.5cm} (2)

In this model, the sending and receiving units, queues, and the processing unit are described as $G_R(s)$ which is to be controlled by the controller $G_c(s)$. Figure 2 shows the system diagram of $G_R(s)$. $G_R(s)$ contains receiving and sending queues; thus, each packet will be delayed $\Delta t$ by
\[ G_R(s) \text{. Meanwhile, the processing unit takes different times to process each packet such that the router, } G_R(s) \text{, can be considered as a first-order inert element which is} \]
\[ G_R(s) = \frac{1}{T_1s + 1}. \tag{3} \]

In (3), \( T_1 \) is a time constant which can be depicted as the average packet arriving interval. In this paper, we assume that each packet arrives following the Poisson distribution; thus, \( T_1 \) is equal to \( 1/\lambda \), where \( \lambda \) is the packet arriving intensity [16]. In most cases, load balancing techniques always try to reduce MLU as much as possible. However, in the model shown in Figure 1, if the reference link utilization is set to very low regardless how much the network maximum capacity is, the packet loss will be very high because the traffic scheduling algorithm in the controller \( G_c(s) \) has to drop much more packets to meet the reference LU. Therefore, the reference LU is set according to the specific network capacity and estimated (not every accurate stream) total network traffic. In this paper, we set it as a constant \( K \). Thus, \( R_{LU}(s) \) can be presented as follows:

\[ R_{LU}(s) = \frac{K}{s}. \tag{4} \]

\( H(s) \) is the Laplace transformation of the feedback element which is used to derive the actual LU by multiplying it with the output packet size \( Y(s) \), the Laplace transformation of \( y(t) \). LU can always be obtained by dividing link traffic, which is accumulated in a short time, by link capacity; thus, \( H(s) \) can be described as an integrating element which is

\[ H(s) = \frac{1}{T_o s}, \tag{5} \]

where \( T_o \) is the average packet leaving interval. In this paper, we also assume that each packet leaves following the Poisson distribution and \( T_o \) is equal to \( 1/\mu \) where \( \mu \) is the packet leaving intensity. Thus, \( G_R(s) \) can also be formulated as \( M/M/1 \) queuing system [17]. \( Y(s) \) is the Laplace transformation of the output packet size. According to the previous works on network measurement and statistics, we assume that the packet size follows Pareto distribution [18, 19]. Hence, the average packet size is

\[ \text{Size}_{\text{pkt}} = \frac{\alpha \cdot \text{Pkt}_{\text{Header}}}{\alpha - 1}, \quad \alpha > 0. \tag{6} \]

According to [20], \( \text{Pkt}_{\text{Header}} \) is not less than 20 bytes. \( \mu \) is the packet leaving intensity; thus, the average output packet size of router interface in a time unit is \( \mu \cdot \text{Size}_{\text{pkt}} \). The time unit here can be defined as one second. In this paper, we use the average output packet size instead of the actual output rate. More complex packet output rate function can be discussed in future works. Thus, \( Y(s) \) is

\[ Y(s) = \mathcal{L} \left[ \mu \cdot \text{Size}_{\text{pkt}} \right] = \frac{\mu \cdot \text{Size}_{\text{pkt}}}{s}. \tag{7} \]

Therefore, the controller \( G_c(s) \) can be designed according to the following equation

\[ G_c(s) = \frac{\mu \cdot \text{Size}_{\text{pkt}}/s}{[K/s - (1/T_o s) \cdot (\mu \cdot \text{Size}_{\text{pkt}}/s)] 1/(T_1 s + 1)}, \tag{8} \]

then, we reduce (8) to

\[ G_c(s) = \frac{\alpha \cdot \text{Pkt}_{\text{Header}}}{\lambda} \cdot \frac{s(s + \lambda)}{K(s - 1 - \mu \alpha \cdot \text{Pkt}_{\text{Header}})/(1 - \alpha)}, \tag{9} \]

then, we have

\[ G_c(s) = \frac{\alpha \cdot \text{Pkt}_{\text{Header}}}{(\alpha - 1) \lambda} (s + \lambda) \times \frac{s}{Ks + (\mu^2 \alpha \cdot \text{Pkt}_{\text{Header}})/(1 - \alpha)}. \tag{10} \]

From (3)–(6), we can see that the system is critically stable (\( s = 0 \) is a pole). Thus, in (10), we can easily see that when \( \alpha \) is in \( (0, 1) \), the controller is stable (there is no pole when \( s > 0 \)), and hence the system is stable. According to (10), the controller should contain three elements which are the proportional element, the first-order differentiating element, and the differentiating element with first-order inertia. This means, if there is a way that the controller can control the output traffic rate following the characteristic depicted in (10) according to the input error LU (i.e., \( R_{LU}(s) \)), the actual LU will always follow the reference LU when the reference LU is set to static or to not to change very frequently. Therefore, we design a packet scheduling algorithm to approximately implement the control characteristic.

2.2. Algorithm. We first describe our algorithm and interpret how it works. Then, we clarify why it can implement the expected control characteristic. Algorithm 1 shows the packet scheduling pseudocode.

The scheduling algorithm first updates the LU of the link on which the packet is to be forwarded according to the route. If the LU is lower than \( K \), the packet will be forwarded normally. Otherwise, the packet will be forwarded on another link based on two conditions: (1) the LU of the other link is lower than the original one; (2) the router connected on the other link is not on the lower depth of the current router. If any of the above conditions is not satisfied, the packet will be dropped if it is a Transmission Control Protocol (TCP) packet or will be forwarded as usual if it is not TCP. Dropping TCP packets will not affect the integration and reliability of communication [21].

In the algorithm, a packet will be rerouted on another link when the LU of the current link is high. This is a way to slow down the increasing speed of LU on current link while speeding up that on the other link which owns lower LU. Hence, the characteristic of differentiating element with first-order inertia is implemented. This characteristic means that the response acts slower than the stimulation in some parts of the element while acting quicker in some other parts. Thus, the packet rerouting here changes the speed of LU varying and hence implements the characteristic. The characteristic of the first-order differentiating element is that the response reflects the gradient of the stimulation. Thus, it is implemented by dropping TCP packets. It is straightforward that dropping TCP packets will decrease the sending rate of source because of the flow control mechanism of TCP [21]. Hence, this actually decreases the packet arriving rate on the
The function Schedule is invoked each time a packet is received. In the pseudocode, pkt is the current processing packet, pkt.len is the length of pkt, link.sizeInT is the total size of received packet in time interval T and to be forwarded on "link", getCap(route) is used to get the link capacity of route, K is the constant reference LU and listLU is the list of LU of each link on the router.

Input: pkt.
Output: pkt, return status.

Init(LU); //LU is init to 0
Init(lastTime); //lastTime is init to 0

Schedule(pkt)
(1) link = getLink(getRoute(pkt));
(2) link.sizeInT += pkt.len;
(3) curTime = getCurrentTime();
(4) if T <= curTime – lastTime then
(5) trfRate = link.sizeInT/(curTime – lastTime);
(6) lastTime = curTime;
(7) sizeInT = 0;
(8) cap = getCap(getRoute(pkt));
(9) LU = trfRate/cap;
(10) endif
(11) if K – LU >= 0 then
(12) return NORMAL; //Forward as usual
(13) elseif (minLU = min(listLU)) < LU then
(14) if the link with minLU that connects the router which is not on the lower depth of the current router then
(15) otherLink = getLink(getRoute(minLU));
(16) otherLink.sizeInT += pkt.len;
(17) link.sizeInT -= pkt.len;
(18) return otherLink; //Output on otherlink
(19) else
(20) Mark the last minLUas temporarily deleted from listLU;
(21) return NORMAL;
(22) endif
(23) elseif pkt.prototype == TCP then
(24) link.sizeInT -= pkt.len;
(25) return DROP_PKT; //Drop the packet
(26) endif
(27) return NORMAL; //Still forward on “link”

Algorithm 1: Pseudocode of packet scheduling.

2.3. Routing Protocol. From the statement 14 of the Algorithm 1, we can easily see that a routing protocol is needed to ensure the link, on which packet is rerouted, not to be connected with node which routes the packet to the current node again (directly or within some hops). Thus, in this paper, we design a routing protocol which includes two phases to ensure that each packet can be routed correctly in WSN. First, each router can obtain the next hop and path cost of each neighbor to the base station by exchanging the routing entries with its neighbors. Second, the neighbors which are closer, but not on the shortest path of current router to the base station, will be put in the candidate set of current router in which the routers are used to share traffic with the router that is on the shortest path. Once each router obtains the candidate set, Algorithm 1 can be performed each time a packet is received.

Generally, most traffic in WSN is generated by sensor nodes and is sent to the base station. Thus, each router only needs to flood its router ID (which can be generally derived
The function $CndDt$ means "Candidate Determination" which is used by each router to derive the neighbors that can be used by Algorithm 1 to share traffic. In the pseudocode, $rtEnt$ is the routing entry, which designates to the base station, of a neighbor; $rID$ is the Router ID of the neighbor which owns $rtEnt$; $rnb$ is the routing table of the current router; $rnb.bs$ is the routing entry which designates to the base station in $rnb$; $rnb.bs.nxhp$ is the IP address of the next hop; $rnb.bs.cost$ is the path cost from the current router to the base station; $rtEnt.nxhp$ is the IP address of the next hop; $rtEnt.cost$ is the path cost from the router associated with $rID$ to the base station.

Input: $rtEnt$, $rID$.
Output: $rID$. // which is in the candidate set

$CndDt(rtEnt, rID)$

1. if $rID == getRouterID(rnb.bs.nxhp)$ then
2. return NORMAL; // This $rID$ is the router which is the next hop of current router on the shortest path to the base station.
3. elseif currentRouterID == getRouterID($rtEnt.nxhp$) then
4. return NORMAL; // This means the current router is the next hop of $rID$ on the shortest path to the base station.
5. elseif $rnb.bs.cost < rtEnt.cost$ then
6. return NORMAL; // This means the cost from the current router to the base station is lower than that from the neighbor $rID$ to the base station.
7. else
8. push($rID$, CandidateSet); // Mark $rID$ as the router which can be used by Algorithm 1 to share traffic that is forwarded from the current router to the base station. The $rID$ is on the upper depth of the current router. That means, when a packet, that is destined to the base station, is forwarded from the current router to $rID$, it will not be routed back to the current router anyway if every router in the network performs Algorithms 1 and 2.
9. return NORMAL;
10. endif

**Algorithm 2**: Pseudocode of candidate determination.

---

in link state routing protocols, e.g., open shortest path first protocol, OSPF) and the routing entry which designates to the base station. When the information is derived in a router, it is flooded and cached in its neighbors and then will not be flooded again. Due to having the number of neighbors, each router can start to calculate the "candidate set" when all of the routing entries, which are destined to the base station, and router IDs of its neighbors are received. Each neighbor can be determined to be candidate or not by Algorithm 2.

3. Simulation

In this section, we first introduce the environment and perform the simulation in different scenarios to evaluate the effectiveness and performance of our approach. Then, we compare and analyze the result of our solution versus others.

In this paper, we use NS2 [22] to simulate all scenarios of which the parameters are shown in Tables 1 and 2.

There are four types of topologies in Table 1. The *randsmall* and *randbig* are pure-random graphs which are generated by GT-ITM [23]. The *powsmall* and *powbig* are Power-Law graphs which are generated by Inet-3.0 [24]. The two types of graphs are used to simulate different network structures in real world. The only difference between *randsmall* (powsmall) and *randbig* (powbig) is the router number which is used for simulating different size of network. Traffic Dist is used to simulate the different traffic source distribution in the real network. Uniform and Pareto are the most regular.
Figure 3: MLU in each topology and traffic distribution.
higher possibility to form the amount of traffic as well as the smaller nodes. This proves that our algorithm can deal with large demands. Compared to other algorithms, such that higher link utilization may be raised on the links of far from others while being bound by many traffic demands in each algorithm. This may be because some nodes are very connected with the sink in most case. In other words, other hubs in the topology affect MLU very little. In some cases, we obtain lower MLU at higher number of traffic demands. Thus, the biggest hub is the sink whenever there is any data. Finally sent to the sink, which is used to gather the sensing information. On average of all simulations, compared to the regular SPF routing (i.e., NONE), Control Theory reduces MLU by 78.4%. Compared to SLUBP, Control Theory reduces MLU by 53.8%, while with nearly the same time complexity.

4. Conclusion and Future Works

WSNs have been widely studied for decades. There are many research areas that have been explored such as service availability and energy efficiency. Load balancing is one of the key techniques which can be used to solve the tradeoff between the performance and the network lifetime. In this paper, we have proposed a novel real-time load balancing technique which only uses the local real-time traffic information and is based on control theory. We have formulated the packet forwarding problem into a classic control model and have derived a controller model. Then, we have proposed an algorithm to implement the controller and have performed the simulation. The simulation results have shown that our algorithm can balance the network traffic in real time and decreases MLU sharply. Our algorithm has reduced MLU by 78.4% compared to the SPF routing and by 53.8% compared to SLUBP which was proposed in the previous work [25].

There is still much work to be done in this topic. (1) To make sure the controller is stable, it is necessary to employ an artificial factor in the model to allow \( \alpha \) to be any value greater than 0. Then, we need to design a new algorithm to implement the controller precisely and to achieve the stability depicted by the model. (2) Designing the specific routing protocol by which the packets distributed by our algorithm will not be routed in loop.

### Notation

- \( R_{LU}(s) \): Reference input: aimed link utilization
- \( E_{LU}(s) \): Error input: difference between \( R_{LU}(s) \) and feedback (the actual LU)
- \( G_c(s) \): Model of controller in each router
- \( G_R(s) \): Model of router
- \( Y(s) \): Output: packet size which is forwarded currently
- \( H(s) \): Feedback element: to derive the actual LU from output
- \( y(t) \): Inverse Laplace transformation of \( Y(s) \)
- \( T_a \): Average packet arriving interval
- \( T_L \): Average packet leaving interval
- \( \lambda \): Packet arriving intensity
- \( \mu \): Packet leaving intensity
- \( K \): The reference LU
- \( s \): Variable in complex domain
- \( \alpha \): Shape parameter in Pareto distribution
- \( Size_{PK} \): Average packet size that a router forwarded
- \( Pk_{Header} \): Minimum size of packet.

### Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.
Acknowledgments

This research is partially supported by the National Basic Research Program of China ("973" Program) (2013CB329101), partially supported by the National Natural Science Foundation of China (NSFC) (61232017, 61102049, and 6116140454), partially supported by the Beijing Natural Science Foundation (4132053), and partially supported by the National Science and Technology Major Project (W13GY00040).

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” IEEE Communications Magazine, vol. 40, no. 8, pp. 102–114, 2002.
[2] K. Akkaya and M. Younis, “A survey on routing protocols for wireless sensor networks,” Ad Hoc Networks, vol. 3, no. 3, pp. 325–349, 2005.
[3] T. He, J. A. Stankovic, C. Lu, and T. Abdelzaher, “SPEED: a stateless protocol for real-time communication in sensor networks,” in Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems, pp. 46–55, Providence, RI, USA, May 2003.
[4] D. Puccinelli and M. Haenggi, “Arbutus: network-layer load balancing for wireless sensor networks,” in Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC ‘08), pp. 2063–2068, Las Vegas, Nev, USA, April 2008.
[5] H. Dai and R. Han, “A node-centric load balancing algorithm for wireless sensor networks,” in Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM ‘03), pp. 548–552, San Francisco, Calif, USA, December 2003.
[6] R. Kacimi, R. Dhaou, and A. L. Beylot, “Load balancing techniques for lifetime maximizing in wireless sensor networks,” Ad Hoc Networks, vol. 11, no. 8, pp. 2172–2186, 2013.
[7] G. Gupta and M. Younis, “Load-balanced clustering of wireless sensor networks,” in Proceedings of the International Conference on Communications (ICC ’03), pp. 1848–1852, Anchorage, Alaska, USA, May 2003.
[8] Y. M. Lu and V. W. S. Wong, “An energy-efficient multipath routing protocol for wireless sensor networks,” International Journal of Communication Systems, vol. 20, no. 7, pp. 747–766, 2007.
[9] A. Dumbrava, R. Kacimi, R. Dhaou, and A.-L. Beylot, “Proportion based protocols for load balancing and lifetime maximization in wireless sensor networks,” in Proceedings of the 9th IFIP Annual Mediterranean Ad Hoc Networking Workshop (MEDI-HOC-NET ’10), pp. 1–8, Juan Les Pins, France, June 2010.
[10] G. Anastasi, M. Conti, M. di Francesco, and A. Passarella, “Energy conservation in wireless sensor networks: a survey,” Ad Hoc Networks, vol. 7, no. 3, pp. 537–568, 2009.
[11] P. Kuila and P. K. Jana, “Improved load balanced clustering algorithm for wireless sensor networks,” in Advanced Computing, Networking and Security, pp. 399–404, Springer, Berlin, Germany, 2012.
[12] K. J. Astrom and R. M. Murray, Feedback Systems: An Introduction for Scientists and Engineers, Princeton university press, Princeton, NJ, USA, 2010.
[13] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” in Proceedings of the 5th ACM International Conference on