An Improvement over RED protocol for Congestion control in Sensor Clouds

Submitted by

Prateek Narendra 12IT50

Nitin Jamadagni 12IT47

Karthik Koralla 12IT33

Submitted to

Dr. Geetha V and Ms. Deepthi L

Department of Information Technology

NITK, Surathkal, Mangalore

In partial fulfilment of the requirements for the course of

Wireless Sensor Networks - IT410

Department of Information Technology

National Institute of Technology Karnataka Surathkal

2015-2016

Abstract

Wireless Sensor Networks (WSNs) and Cloud computing are two emerging technologies, respectively, in the networking and computing fields. Sensor Cloud integrates WSNs and Clouds into an extended form of cloud infrastructure that manages sensors scattered throughout WSNs. Networking plays a crucial role in Cloud computing and becomes a particularly challenging issue in sensor Clouds due to resource limitation of WSNs as the underlying network of the Cloud infrastructure. Therefore, data transmission from sensors to servers in sensor Clouds must be carefully controlled to avoid network congestion. We have studied Random Early Detection-Based (RED-based) congestion control for data transmission in sensor Clouds. Then we propose an improved RED algorithm that meets sensor Cloud networking requirements better and develop a queueing model coupled with analysis technique to evaluate the performance of the proposed congestion control scheme. We conduct basic simulations to evaluate performance of Improved RED compared with the regular RED algorithm. Our analysis and simulation results demonstrate that the proposed congestion control achieves better drop rate performance in comparison with the standard RED for data transmission in sensor Clouds. The easily implemented control mechanism and low complexity of our proposed algorithm make it more suitable for sensor Clouds.

Contents

List of Figures

1 Introduction

In the past decade, Wireless Sensor Networks (WSN) have been widely deployed in various applications, including industrial automation, environmental monitoring, transportation, and health-care, as a promising technology for enabling a broad spectrum of novel and attractive solutions. WSNs have become an indispensable component of the future information infrastructure upon which a large number of computing applications can be developed. In order to simplify system operation and maintenance as well as to reduce costs, WSNs must become an infrastructure that is capable of supporting various applications for multiple users concurrently, instead of having to roll out individual networks for specific purposes.

Cloud computing has emerged as a new computing paradigm that enables infrastructure resources to be visualized and provisioned as services on demand. Inspired by such a new service model sensor Clouds have been evolved as an extended form of Cloud computing to manage sensors scattered in WSNs. Sensor Clouds integrate WSNs and Cloud computing, two promoting technologies in the networking and computing domains, and offer a novel platform that stimulates development and deployment of advanced WSN-based sensor Cloud applications. Networking systems play a crucial role in Cloud computing and have a significant impact on performance of the services that can be offered by Clouds. Recent study on Cloud performance has indicated that in many cases data communications in Cloud infrastructure form a bottleneck that limits Clouds from supporting high-performance applications.

High-performance networking becomes an even more challenging issue in sensor Clouds, where a large amount of data collected by sensors need to be transmitted across WSNs to servers for processing in order to support various applications. Due to the limited bandwidth in typical WSNs, data transmissions in sensor Clouds must be carefully controlled to avoid network congestion. Transport control protocols are used to mitigate congestion and reduce packet loss, to provide fairness in bandwidth allocation, and to guarantee end-to-end reliability. A variety of works have been reported in the literature for revising traditional TCP and UDP protocols to meet WSN requirements.Active queue management is to complement the work of end-to-end protocols such as TCP in congestion control so as to increase network utilization and limit packet loss and delay. Random Early Detection (RED), a typical active queue management scheme for network congestion control, offers a promising congestion control mechanism for data transmission in sensor Clouds. Although RED is well studied and widely applied in computer networks, the existing research mainly considered regular networking scenarios instead of WSNs for data transmissions in sensor Clouds.

Common constraints related to the nature of WSNs have to be tackled. Network nodes in WSNs typically have very limited computational abilities, memory space, and power resources. Regular RED as an active queue management scheme requires a relatively complex procedure for processing each incoming packet, including calculating weighted average queue length, predicting a drop probability, and making packet drop decision. Such a procedure may become a burden to sensor network nodes that limits data transmission performance and shortens network lifetime. Therefore, it becomes necessary to investigate appropriate application of RED-based control scheme in sensor Clouds for achieving comparable networking performance with low control overhead. Our main areas of work in the project is that

1. Analyze RED protocols for its drawbacks and suggest an improvement in the protocol based on the observations

2. Design, Implement and Compare observations for the improved RED algorithm over a simulated WSN scenario

2 Related Work

Research efforts on queue management have been made to avoid network congestion in the last decades. The current queue management mechanisms can be classified into two categories: Passive Queue Management (PQM) and Active Queue Management (AQM). Drop tail (DT) is a typical PQM,in which a maximum queue length is set to accept

packets. When the queue is full, the packets will be discarded until the queue length is below the maximum threshold, and then the newly arriving packets will be re-accepted. Nevertheless, this approach is limited by the problem of lock-out and global synchronization. Lock-out may cause long latency and global synchronization rejects most of packets in a burst,thus degrading service performance. Therefore, DT is not suitable to control data transmissions for supporting high-performance sensor Cloud applications. AQM has been proposed to proactively detect indicators of network congestion and drop or mark packets randomly in order to overcome the limitation of PQM. RED is a widely used active queue management scheme that was first proposed by Floyd and Jacobson. RED is expected to continue in foreseeable future as a typical AQM for network congestion control. Standard RED calculates drop rate using the average queue length and packets will be dropped probabilistically depending on threshold settings in the queue.
There is a large number of research works related to the subject. For example, focused on the parameter settings to improve the service performance. In order to enhance the robustness of RED, researchers developed some algorithms based on RED such as Adaptive RED , Blue , and Weighted RED .

The aforementioned works on RED algorithms are mostly dedicated to RED parameters adjustment for performance improvement. These works mainly considered regular com- puter networks, for example, the Internet backbone, as the main networking scenarios. The underlying networking platform of sensor Cloud consists of WSNs, which have many special features that distinguish them from traditional networking scenarios and bring in new challenges to congestion control. Various technologies have been developed to address this issue.

3 Methodology

As service computing technologies bridge the gap between sensor applications and WSN infrastructure, there should be a data processing platform that processes the raw data collected in WSNs. An illustrative system framework is shown in Figure 1, which consists

of three layers: sensing layer, data processing layer, and service layer.
The sensing layer contains sensor nodes for collecting data from the physical world. is layer only brie y processes the large amount of collected data and then delivers them to the data processing layer.
The servers on the data processing layer perform computation functions to process collected data and store the obtained results as information resources in order to provide services requested by applications via the service layer. The service layer provides a common upper-level interface to sensor applications for utilizing sensor Cloud infrastructure while hiding the details of data collection, transmission, and processing in the sensor Cloud infrastructure, thus facilitating service provisioning to support various sensor applications deployed upon the sensor Cloud.
Network links in WSNs of a sensor Cloud often have limited transmission bandwidth and servers in the processing layer also have infinite capacities. Therefore, congestion control with active queue management is important in sensor Clouds for meeting the performance requirements of service provisioning.

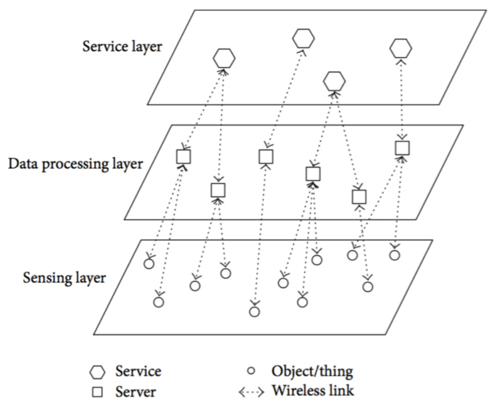

Figure 3.1: An illustrative system framework for sensor Cloud networking

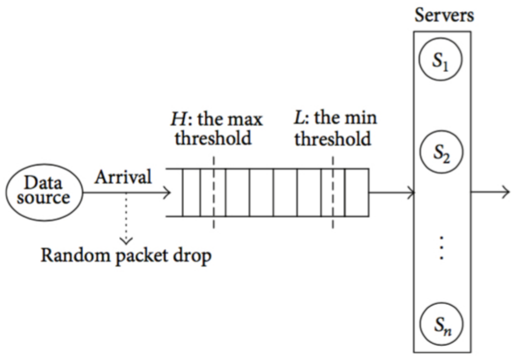

Figure 3.2: An improved model of RED-based congestion control for data transmission in sensor Cloud.

Algorithm 1 Improved RED Algorithm

for *eachpacketarrived* **do**
    calculate instantaneous queue length k;
    if $k < L$ **then**
        add packet to queue;
        $k++$
    end if
    if $L <= k <= H$ **then**
        calculate the packet dropping probability
        $p_b = \frac{f_m \times (k-L)^2}{(H-L)^2}$
        $p_a = \frac{p_b}{(1-flag \times p_b)^2}$
        drop the packet with probability p_a
    end if
    if The current packet is dropped **then**
        $flag = 0$
        $count++$
    else Add this packet into queue,$k++$
    end if
    if $H <= k <= K$ **then**
        Drop the packet with probability p_{max}
        if The packet is dropped **then**
            $flag = 0$
            $count++$
        else
            Add the packet P into queue
            $k++$
        end if
    end if
    if $K < k$ **then**
        Drop packet
        $count++$
    end if
end for

3.1 Software Requirements

Some technologies used in development include:

1. Network Simulator v2.35

2. NAM

3. GNUPlot, Tcl, C++

4 Implementation

The project was carried out in 2 different phases -

1. Modification of the existing RED protocol in Network Simulator

2. Creating a topology and comparing the drop rates of RED and Improved RED

Natively, Network Simulator implements RED protocol and its variation like Adaptive RED, Blue etc. These are implemented in the form of extensive C++ and Tcl Scripts. These scripts were modified accordingly and recompiled , effectively adding the IRED algorithm to the Network Simulator. Thus, we were able to simulate the improved RED queue within our built network topology.
In our topology, 20 nodes were created, each with a duplex link between them and CBR Rate of 5Mbps, each having queue of type RED or Improved RED. The rate of data flow was set to 10Mbps with a delay of 20ms.
For transmission, we used 3 different types of protocols :

1. TFRC

2. TCP

3. Binomial

Runs were randomized using a random seed. The results were captured as a live simulation on NAM and the dumps were used as the data source for plotting the comparison of drop rates between RED and improved RED protocols.

5 Results and Analysis

In this section we present a comprehensive evaluation of the performance of our algorithm with respect to RED algorithm.
We performed the experiment on a Network Simulator v2.35 on MacBook Pro running OSX 10.11.4 with 8GB RAM, 4 CPU cores and Intel Processor running at 2.7GHz. The graphs below are self explanatory for the drop rates achieved by both algorithms.

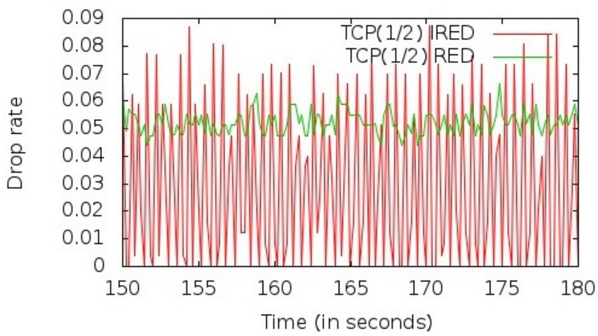

Figure 5.1: Comparison of drop rates for TCP protocol (1/2)

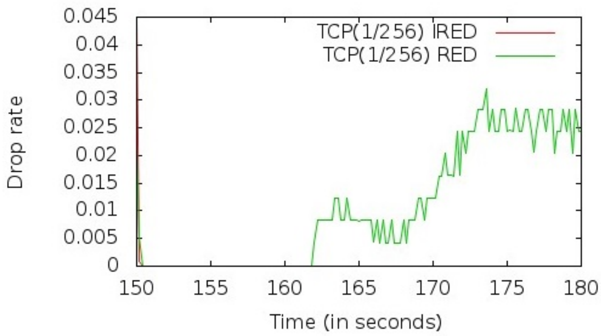

Figure 5.2: Comparison of drop rates for TCP protocol (1/256)

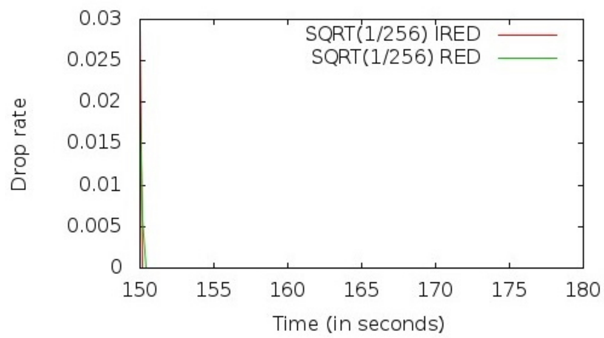

Figure 5.3: Comparison of drop rates for SQRT protocol (1/256)

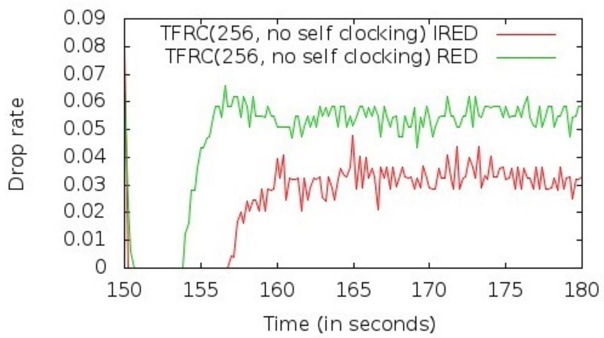

Figure 5.4: Comparison of drop rates for TCP protocol (1/256) with no self clocking

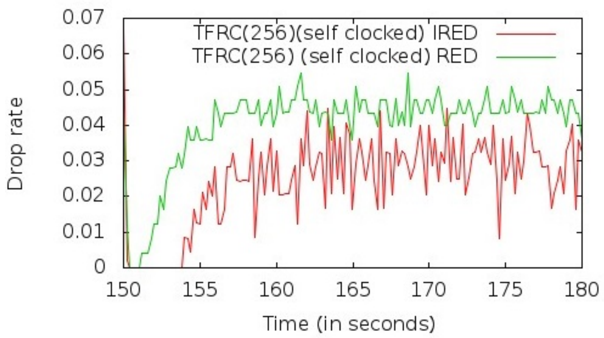

Figure 5.5: Comparison of drop rates for TCP protocol (1/256) with self clocking

We can clearly observe from the results that drop rates achieved by improved RED protocol is much lower than native RED algorithm. However, due to single machine constraints, we were unable to evaluate a very large WSN scenario including thousands of WSN nodes.

The throughputs achieved for the model scenario have been comparable.

6 Conclusion and Future Work

6.1 Conclusion

We can clearly conclude that there is scope for improvement in the RED protocol. Particluarly, if specific WSN scenarios are taken, there is scope for different kind of RED queues that can be implemented to suit the scenario, given that the drop rates is very low in our given scenario of 20 nodes, we can conservatively extrapolate this to a real world WSN deployment. Thus, our effort to improvise on the existing RED protocol has been successfully demonstrated with the improved RED protocol.

6.2 Future Work

In the future, we plan to extend the project to those scenarios where base stations or cluster head nodes are connected to the internet. Congestion control in such scenarios is much important as there exists a single TCP/IP connection.

We also plan to extend the improved RED algorithm to Adaptive RED algorithm , where probabilities and thresholds are dynamically changed to accomadate varying data rates at the sensor nodes.

References

[1] Sally Floyd and Van Jacobson - Random Early Detection Gateways for Congestion Avoidance - IEEE/ACM Transactions on Networking , August 1993