Computer Networks 54 (2010) 1672–1682

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Comparing the results from various performance models of IEEE 802.11g DCF

P.S. Kritzinger [a,*,1], Henry Msiska [a], Tino Mundangepfupfu [a], Paolo Pileggi [a], Andrew Symington [b]

[a] Data Network Architectures Group, Computer Science Department, University of Cape Town, South Africa
[b] Computing Laboratory, Oxford University, England, UK

ARTICLE INFO

Article history:
Received 13 April 2009
Received in revised form 18 September 2009
Accepted 3 January 2010
Available online 20 January 2010
Responsible Editor: T. Tugcu

Keywords:
Simulation modelling
IEEE 802.11
Medium access control (MAC)
Distributed coordination function (DCF)
WLAN
Wireless test bed

ABSTRACT

IEEE 802.11 DCF is the most widely-used CSMA/CA access control mechanism. Recent analytic performance models for DCF have received acclaim for both their simplicity and reported accuracy. Most of these models share the assumptions of full single-hop connectivity among all stations, that DCF back-off may be modeled as a Markov process and that the network is saturated with traffic. In order to verify the accuracy of existing analytic models we developed a discrete-event simulator to record the performance of the DCF protocol and ensure that every detail of the standard is represented. Simultaneously we set up a hardware test bed to measure the same performance metrics in an environment that makes none of the simplifying assumptions of either the analytic models or the simulation. In the test bed, as in the simulator, we used the same physical parameter settings prescribed by the standard. As is the case for the analytic models we used, we subjected the simulator and the test bed to the same saturated workload for both basic and RTS/CTS access modes. Finally, we also implemented a non-saturating Markov Modulated Arrival Process (MMAP) workload model for our simulator to test the performance of DCF subject to more realistic internet traffic conditions. We describe both the simulator and the test bed in some detail in order to testify to the accuracy and detail of our results. The results show that the analytic models are mostly pessimistic for small numbers of nodes and optimistic for larger numbers of nodes. The performance measurements from the test bed, in turn, indicate that the simulation results are similarly optimistic when large numbers of nodes are concerned. Since the test bed uses an error-prone wireless channel, this latter result is, in principle, not surprising. The rate of deterioration in the actual performance is however something that is not widely known and is much more rapid than analytic models would suggest.

1. Introduction

The popularity of wireless networks is gaining momentum due to their cost effectiveness and ease of installation. In particular, IEEE 802.11 Wireless Local Area Networks (WLANs) consist of several nodes that connect to the Internet through a central Access Point (AP). In this infrastructure topology all nodes must be situated within the wireless footprint of the AP. Since several nodes share the same wireless channel, it is clear that only one node can, at any one point in time, transmit successfully on that channel. Regulating channel access efficiently is therefore pivotal to maintaining a favorable performance. The IEEE 802.11 standard [1] defines several channel access control mechanisms to achieve this, the most widely-used being the Distributed Coordination Function (DCF).

In recent years, a great deal of research has been conducted to elucidate the effect of CSMA/CA protocols on

* Corresponding author. Tel.: +27 833035445; fax: +27 216503551.
E-mail address: psk@cs.uct.ac.za (P.S. Kritzinger).

1 The Data Network Architectures (DNA) research group is funded in part by National Research Foundation (NRF), Telkom Siemens and Technology and the Human Resources for Industry Programme (THRIP).

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2010.01.003

aggregate network performance. Analytic models, simulation and hardware prototypes have all been used to measure such performance. Analytic models are mostly preferred because of their efficiency of computation. However, to preserve analytic tractability one is forced to impose unrealistic assumptions. Although analytic models for DCF have been shown to produce reasonably accurate results, they tend to make very restrictive assumptions about the characteristics of the workload.

Hardware experiments are rarely carried out, since they are costly and require more time and effort in terms of deployment and maintenance. A hardware experiment is effectively a small-scale version of the actual system, which is deployed in a controlled environment. Despite their limited scalability, such experiments can produce more accurate results since they reflect the nature of a real network most accurately. Examples of wireless network test beds are the Orbit Laboratory [2], MIT Roofnet [3] and the DNA Test bed [4]. Such networks are costly and time consuming to set up and researchers therefore often turn to simulation.

For a simulation one can either apply existing simulator platforms such as the OPNET Modeler®[5], ns2 [6], QualNet [7] or OMNET++ [8] or develop one's own simulator by implementing the simulation engine and components in a language such as Java. The OPNET Modeler®Wireless Suite, as one instance, provides high fidelity modeling, simulation, and analysis of a broad range of wireless networks. But these simulators are complex, general-purpose software suites and it is seldom clear which details of the network stack are being modeled and where all the associated parameters may be found. More controversially, some of these systems are commercial products and for proprietary reasons or otherwise do not make clear [9]

- whether one is assured that the simulation has stabilized before sampling,
- what the sample sizes are
- or whether sampling is done to ensure identically, independently distributed (IID) variables.

In a controversial paper Cavin et al. [10] illustrated these points by showing the deviation in results among several widely-adopted simulators, such as OMNeT++ and NS-2 for a sample mobile ad-hoc network experiment. Statements such as "... users are responsible for verifying for themselves that their simulations are not invalidated because the model implemented in the simulator is not the model that they were expecting..." [6] are not very encouraging when one is conducting a scientific study. Nevertheless, these general simulation platforms and libraries are obviously useful as witnessed by the many published studies that use them.

However, there are others such as the authors and Bianchi and Tinnirello [11] who prefer the second option. They

- sacrifice the convenience of these software platforms and at the same time
- spend much time delving down into the minute system detail of the system being modeled to ensure that these are represented in the simulation.

Simultaneously it is then possible to measure and record exactly those performance values relevant to the study. In contrast to the use of general-purpose simulation platforms, we call this *deep simulation*. Deep simulators make no claim to be general or to replace the existing general simulation platforms.

By the very nature of the work reported upon here, the authors had to develop their own deep simulator. We calculated normalised aggregate throughput, channel efficiency and packet delay for a modern IEEE 802.11g network using the simulator and list all of the parameters used for the experiments. We compare the simulation results to those from another deep simulator developed by Bianchi [11] and his colleagues, to the results from several analytic models as well as to measurements taken from the DNA test bed. We show that the results from our test bed experiments deviate significantly from the analytic models. It turns out that the analytic models are pessimistic for small numbers of nodes and optimistic for larger numbers of nodes. Finally, to model a more real-world environment, we used a 6-state Markov Modulated Arrival Process (MMAP), rather than a saturated workload, to model the network traffic.

2. Distributed Coordination Function

The DCF protocol dictates that time at each station is divided into fixed length slots of duration σ microseconds. A node wishing to transmit a frame first monitors the channel for a Distributed Inter Frame Space (DIFS) period. If the channel is idle, the node backs off for a random number of slots picked uniformly from an integer interval called the contention window. The length of the contention window starts at CW_{min} and doubles in length after every failed retransmission, up to some maximum length CW_{max}. The back-off counter is decremented every σ period until it reaches zero when transmission is attempted. This specific action avoids collisions, since other nodes may have been monitoring the channel for the same purpose.

If the channel is sensed busy during back-off, the back-off counter seizes and is only reactivated when the channel has been sensed idle for a full DIFS period. When the back-off counter reaches zero the node retransmits the frame. If two or more stations transmit simultaneously, a collision occurs at the receiver. To resolve a collision, all nodes involved in the collision restart their back-off, while all other nodes in the neighborhood counters freeze for a period called the Extended Inter Frame Space (EIFS). Thus, colliding nodes have a greater probability of medium access for this post-collision period. All frames are acknowledged within a Short Inter Frame Space (SIFS) period. If the sender does not receive an acknowledgment it either reschedules transmission or drops the packet, depending on its retry count. The microsecond values for σ and the Inter Frame Spaces are determined by the modulation scheme employed in the PHY and are given for pure IEEE 802.11g in Table 1. The standard specifies two channel access schemes for the DCF: Basic Access (BA), and Request-To-Send/Clear-To-Send (RTS/CTS).