Simulation Framework for Wireless Internet Access Networks

Hyoung-Kee Choi and Jitae Shin

The School of Information and Communication Engineering,
Sungkyunkwan University,
Suwon, Korea 440-746
{hkchoi, jtshin}@ece.skku.ac.kr

Abstract. In many fields of engineering and science, researchers and engineers use computers to simulate natural phenomena rather than conducting experiments involving the real system. With the power of today's computers, simulation provides an easy way of predicting the performance of a prospective system or comparing several alternatives at the system design stage. In this paper, we present a simulation framework specifically designed for wireless Internet access networks. This simulation framework is designed for the three protocol layers, HTTP, TCP/IP and the link-layer protocol, and can be configured independently in each of these layers. The time-driven nature of the framework allows us to observe the diurnal changes of the system in the simulation, which in turn makes it possible to evaluate the statistical properties of the system.

1 Introduction

In recent years, Internet technology has emerged as the major driving force behind new developments in the area of telecommunication networks. The volume of packet data traffic has increased at extreme rates. In order to meet these changing traffic patterns, more and more network operators are adapting their strategies and are planning to migrate to IP-based backbone networks. Meanwhile, mobile networks face a similar trend of exponential traffic increase and growing importance to users. Recently, in some countries, such as the Republic of Korea, the number of mobile subscriptions has exceeded the number of fixed lines.

The combination of both developments, the growth of the Internet and the success of mobile networks, suggests that the next trend will be an increasing demand for mobile access to Internet applications. It is therefore increasingly important that mobile radio networks support these applications in an efficient manner. Thus, the mobile radio systems currently under development include support for packet data services. For instance, General Packet Radio Service (GPRS) is the wireless packet data service on the Global System for Mobile (GSM). New wireless packet data services have also been introduced for the Wireless Local Loop (WLL) and CDMA2000 systems, which operate in parallel to the existing wireless circuit voice services.

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3515, pp. 453–460, 2005.

When the new wireless packet data systems were designed, a simulation was used to predict the performance of the prospective system or to compare several alternatives [1],[2],[3],[5],[7],[8]. Hence, it is important to have a reliable simulation framework which accurately predicts the performance of the future system. A number of simulation frameworks have been introduced for network systems. Of these, *ns-2* (Network Simulator), ssf (Scalable Simulation Framework) and Opnet are the most popular tools for network simulation. These simulators are generally considered to be the best tools available and the models on which they are based have been thoroughly validated by a number of user groups. However, in the case of wireless network simulations, the choice of one of these tools might not be optimal, since they were all developed for general purpose networking simulations.

In a previous study, we reported the development of a behavioral Web traffic model [4]. Since then, this model has been used in at least three studies to evaluate the performance of wireless networks [5],[7],[8]. However, these studies did not include certain key points such as how the model was adapted to satisfy the requirements of the study, how the different stacks of the protocol interacted with one another to produce the right simulation results, and so on. In this paper, we provide more detailed and up to date information on our model and the associated simulation framework, which are designed to be used in wireless internet access networks.

2 Proposed Model

We characterize the HTTP and TCP layers as well as an underneath link-layer protocol of interest based upon the typical transactions of the individual protocol. We examined the transactions associated with the retrieval of a single Web page and selected a set of primary parameters and secondary parameters in each layer. The primary parameters can be used to define the Web traffic model, and the secondary parameters help to understand the behavior of the Web traffic. In the following discussion, we use **boldface** to indicate a parameter.

2.1 HTTP Model

We characterize Web traffic based upon a typical transaction of a Web page in HTTP, as shown in Fig. 1. The number of objects refers to the total number of objects in a Web page. Nine objects are in the Web page shown in Fig. 1. **Number of in-line objects** is eight, and there is also one main object. We count only those objects that need to be downloaded. If an object is cached, the browser uses the object in the local cache after validating the object.

A new Web page is generated immediately after the expiration of the viewing period. HTTP alternates between the ON and OFF states, in which the ON state represents the activity of the Web page and the OFF state represents the silent time after all the objects in the Web page have been retrieved. The durations of the ON state and the OFF state correspond, respectively, to **on-time** and

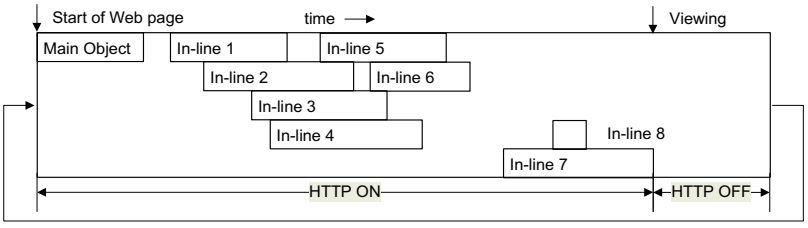

Fig. 1. A typical transaction of a Web page in HTTP

viewing time. The ON state can be further split into the successive TCP connections used to deliver individual objects. The five rows in Fig. 1 represent distinct connections.

We refer to the connections in rows 1, 2 and 4 as keep-alive connections, because multiple objects are delivered in one connection. In Fig. 1, we consider in-line objects 1, 5, 6 and 8 as being delivered on the keep-alive connections. **Keep-alive ratio** is calculated by dividing the number of objects delivered on the keep-alive connections by the number of objects (the number of in-line objects plus one) for a Web page. **Keep-alive ratio** in the above example is 0.44 (4/9). The inactive period between in-line objects 6 and 7 is denoted as **in-line inactive time**. In particular, the inactive period between the main object and in-line object 1 is required in order to parse the HTML code, and is denoted as **parsing time**. A Web server identifies a requested object based on the HTTP request header. We denote the size of the HTTP request header as **request size**.

2.2 TCP Connection Model

We characterize TCP based upon a typical transaction of a TCP connection used to retrieve a Web object, as shown in Fig. 2. At the beginning of the connection, the client and server exchange control segments (SYN) to synchronize with each other. It takes a single round-trip time (RTT) to exchange SYN segments (see the 0th period in Fig. 2). Once the synchronization process is completed, the client sends a request for an object (REQ in Fig. 2) to the Web server.

TCP alternates between inactive and active periods of transmitting data segments. After a TCP sender transmits an entire window-size worth of data segments in a burst, it pauses until the first ACK corresponding to the burst returns (see first and second periods in Fig. 2). This is because the window size is still too small for the pipe to be completely filled. Then, TCP starts the next burst after adjusting the window size. As the window size increases, the inactive period decreases.

Based upon the bursts in a TCP connection, we characterize TCP by: (1) defining the period between the starts of adjacent bursts and (2) measuring the number of data segments transmitted and the time spent in this period. Let us denote the period between the starts of adjacent bursts as a window epoch or

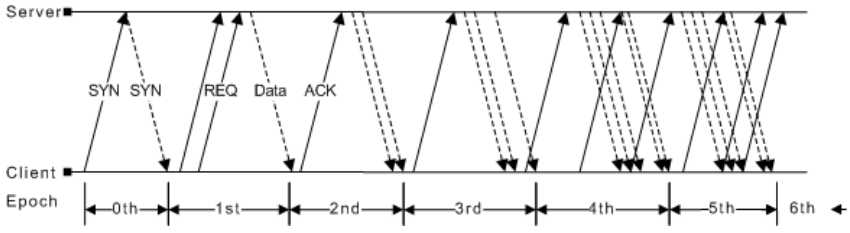

Fig. 2. A typical transaction of a TCP connection transferring a Web object

an epoch. Let us denote the number of data segments transmitted and the time spent in an epoch as **number of segments in an epoch** and **epoch time**, respectively.

2.3 Link-Layer Protocol Model

A number of wireless protocols can be combined in this proposed simulation framework, in order to measure their performance. To help understand how one can use the proposed simulation framework, we illustrate one typical simulation procedure with a MAC protocol in a satellite network. This MAC protocol was developed by the author and the details of the MAC protocol can be found elsewhere [5].

Remote stations under consideration are Web clients using the TCP stack as their primary transport protocol. The remote stations send a request to the hub to gain access to the physical channel. The hub receives the request and schedules it according to the centralized priority reservation (CPR) protocol. The physical channel is divided into a number of forward and return links. Forward links are generally larger in capacity than return links, as they carry more data to the remote terminals than the return link carries to the hub.

When an IP packet is generated at the remote station, it is passed to the MAC layer. Upon receiving a packet, the MAC protocol divides it into data frames. Before transmitting data frames, the remote station sends a request for transmission to the hub. This request is sent on a contention-basis. A collision may occur between different remote stations having a request to make at the same time. Provided that no collision occurs for the request, the hub acknowledges the request immediately. If a collision occurs when the request is being sent, the remote station will not receive an acknowledgment from the hub after the round-trip delay and will then attempt to retransmit the request. A contention resolution algorithm (CRA) is used to resolve the collision.

3 Traffic Generation

The complete model of Web traffic is a combination of three models: the HTTP model, the TCP model and the model of the link-layer protocol of interest. These three models interact with each other to form a total model that encompasses all

Web traffic. The statistics of the parameters and their probability distributions can be found elsewhere [4],[5],[6].

3.1 HTTP Layer

Our model in the HTTP layer simulates an ON/OFF source. At the beginning, the traffic corresponding to the main object is generated and is delayed for the period of **parsing time**. During this period, the Web browser fetches the main object and parses number of in-line objects as well as the page layout. The HTTP model, however, generates the value of **number of in-line objects** from the best-fit distribution and waits for the expiration of **parsing time**.

After the start of one in-line object, there is a delay before the start of the next. The first in-line object starts after the expiration of **parsing time**. The second in-line object does not wait until the first in-line object finishes, but starts one **in-line inter-arrival time** after the start of the first. Subsequent in-line objects continue to start until the number of in-line objects started equals **number of in-line objects**. In the model, depending upon **in-line object size** and **in-line inter-arrival time**, the number of outstanding connections will vary. Frequently, **in-line inter-arrival time** is less than the duration of the connection, which is mainly determined by **in-line object size. Hence, the model indirectly simulates the parallel downloading of in-line objects**. After all of the objects have been transmitted, the model is silent for a period of time corresponding to **viewing time**. After the expiration of this time period, the model starts to generate a new Web page.

The Web caching model influences the final model through main object size and in-line object size. Due to frequent changes that they undergo, the main objects are fetched most of the time, rather than being cached. The HTTP object size becomes zero, except for that of the main object if it is destined to be cached. Otherwise, the sizes of both HTTP object types are generated from the distribution.

3.2 TCP Layer

For the complete model of Web traffic, the TCP model relies on the HTTP model to obtain essential information regarding a connection. The TCP model obtains the object size from the HTTP model. In addition, a real connection can be a keep-alive connection, delivering more than one object. The HTTP model determines whether a given connection is a keep-alive connection and the elapsed time between objects.

At the beginning of a connection, the client in the model exchanges a SYN segment with the server, mimicking the three-way handshaking procedure. After the three-way handshaking procedure, the client enters the first epoch by sending a request segment. At this point, the HTTP model informs the TCP model of **request size**, **main object size** and **in-inline object size**. The model calculates the total number of segments in the connection by dividing the object size by an MSS of 1,460 bytes. The number of segments in the first epoch and

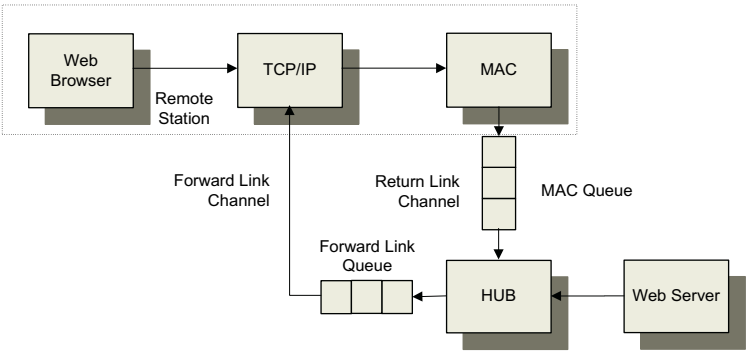

Fig. 3. Software Implantation of the link-layer model

the time the first epoch lasts is determined from the distributions of number of segments and epoch time given epoch number one.

In a given epoch, the TCP model generates a burst of segments followed by an idle period until the next epoch starts. At the end of each epoch, the model checks whether the server has transmitted enough segments for the given object size. The model proceeds to the next epoch, as long as the cumulative number of transmitted segments is less than the total number of segments. At the last epoch, when the model finishes downloading the current object, the HTTP model informs the TCP model of its decision as to whether the connection should remain open for the delivery of the next object or should be closed.

3.3 Link Layer

Although it is the TCP model that determines when to send packets, it is the link-layer model that determines how to send packets in the wireless channel. For this particular illustration of the simulation, we set the return-link bandwidth and forward-link bandwidth to 32 kbps and 45 Mbps, respectively. The RTT between the remote station and the hub through the satellite was set to 0.5 seconds. The number of remote stations is varied, in order to observe the changes in throughput and load induced by changing the number of stations.

Once a packet is available at the remote station, the TCP model informs the link-layer model of the packet size. The link-layer model segments the packet into frames of the proper size, as determined at the protocol design stage. For individual frames, the link-layer model simulates the MAC protocol, in order to obtain access to the wireless channel. Once the remote station has secured the right to access the wireless channel, it may deliver the frame to the hub.

Due to the time-driven nature of the framework, the clock in the simulation increments with the fixed size of an interval. This time-driven simulation enables us to observe the longitudinal changes of the system. In this way, it is a lot easier to evaluate the statistical properties of the system being examined. A great deal of care needs to be taken in deciding the granularity of the interval. With a small

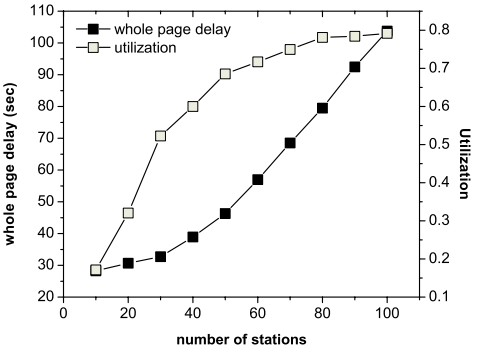

Fig. 4. Delay vs. number of stations and channel utilization vs. number of stations

granularity, the system can be observed in detail, but the simulation is likely to take too long. In our design, the incrementing interval is equivalent to the time delay required for transmitting one frame over the wireless channel.

Frames arriving at the hub are reassembled in order to be transmitted over the wire line toward the Web server. Since our primary interest is the system behavior in the wireless channel, we do not specifically implement the mechanics of the wire line in the simulation. Instead, the mechanics of the wire line are replaced by the statistics collected in the trace. In this way, we can reduce the complexity of the simulation, without sacrificing the accuracy.

The procedure used for the return-link channel is also applied to the forward-link channel, with two exceptions, namely the bandwidths are different and the MAC protocol does not exist in this channel. The packets arriving at the remote station are processed in the link-layer protocol and then passed successively to the TCP and HTTP layers. From the perspective of the simulation, this is the end of a single round trip for the packet. The next round trip is initiated by the dynamics of the upper layer protocols. However, the procedure used for the next round trip is the repetition of the current round trip.

4 Discussion

Among the numerous measurements that can be obtained from the simulation, the most interesting ones are the delay and the throughput with different loads. Fig. 4 shows a plot of the delay and the throughput versus the load. As the number of users (load) increases, saturation starts to occur when the number of users is in the range of 40 to 50. In addition to this result, we can also extrapolate that the delay varies linearly with the number of remote stations. The feedback nature of the Web browsing application (HTTP) limits the number of outstanding packets, so that at no point does the simulation collapse; the response time just becomes slower.

For simplicity, we assume that the capacity in the forward-link channel is infinite, so that we may not implement the mechanics of the forward-link channel in the simulation exclusively. We refer to this type of simulation as an open-loop simulation. The reason for this is that the bandwidth of the forward-link channel is so big that outstanding frames would not experience any delay in the forward-link queue shown in Fig. 3. As a result, this assumption does not affect the overall performance of the simulation, rather the complexity of the simulation decreases significantly.

It is preferable to obtain a quick response from a given simulation. At the same time, a stead-state response is also required. However, this is not an easy task, because the simulation experiences a transient state in the beginning. For instance, the queues in the simulation are completely empty in the beginning and take a certain amount of time until they reach a steady state. Any conclusions drawn from the transient state may undermine the accuracy of the simulation. Hence, the simulation must be long enough to allow a steady state to be reached.

5 Conclusion

We proposed a simulation framework for wireless Internet access networks. This framework was developed in three layers, the HTTP, TCP/IP and Link-layer protocols. Because of this layered structure, one can not only examine the behavior of the system being examined at each layer, but also observe the overall behavior of the system within the combined layers. Because the proposed framework has a time-driven nature, one can observe the diurnal changes of the system at regular intervals, which is very important when it comes to evaluating the statistical properties of the system.

References

1. Kalden, R. and et al.: Wireless Internet Access Based on GPRS, *IEEE Personal Communcation Magazine*, April 2000.
2. Brasche, G. and Walke, B.: Concepts, Services, and Protocols of the New GSM Phase 2+ GPRS, *IEEE Communcation Magazine*, August 1997.
3. Cai, J. and Goodman, D.: General Packet Radio Service in GSM, *IEEE Communcation Magazine*, October 1997.
4. Choi, H. and Limb, J. O.: A behavioral model of Web Traffic, In *Proceedings of the IEEE ICNP '99*, October 1999.
5. Choi, H. and et al.: Interactive Web service via satellite to the home, *IEEE Communcation Magazine*, March 2001.
6. Choi, H. and Copeland, J. A.: Modeling the behavior of TCP in Web traffic, In *Proceedings of the ICOIN '05*, January 2005.
7. Mohangy, B. and et al.: Application Layer Capacity of the CDMA2000 1xEV Wireless Access System, In *Proceedings of the World Wireless Congress*, May 2002.
8. Staehle, D. and et al.: QoS of Internet Access with GPRS, *Wireless Network*, May 2003.