JITTER-AWARE TIME-FREQUENCY RESOURCE ALLOCATION AND PACKING ALGORITHM

Huan Yao, Thomas Royster IV, Jeffrey McLamb, Mehmet Mustafa, Navid Yazdani [1]

MIT Lincoln Laboratory
Lexington, MA

ABSTRACT

One of the main components of the next generation protected military satellite communication systems is Dynamic Bandwidth Resource Allocation (DBRA). A centralized DBRA algorithm on the satellite dynamically grants terminals time and frequency resources as their traffic demands and channel conditions change, leading to significant increase in the overall system throughput.

This paper address one potential issue associated with DBRA for the satellite uplink channel, which is a Multi-Frequency Time-Division Multiple Access (MF-TDMA) channel. As DBRA dynamically assigns time-frequency slots to terminals, there may be uneven temporal gaps in the assignment if special care were not taken. When this happens, even though average rate demands can be met, applications may experience larger than desired delay and jitter, which may reduce the quality of certain voice and video applications.

This paper presents a novel algorithm for allocating and packing time-frequency slots in a jitter-aware fashion by using groups of evenly spaced slots. The achievable delay and jitter performance is evaluated using an OPNET simulation.

INTRODUCTION

A satellite system employing multi-frequency time-division multiple access (MF-TDMA) must efficiently allocate its available communication resources so that many terminals can be supported. The satellite grants resources to the terminals periodically, in a time period that is known as an epoch. Each terminal receives an assignment consisting of a center frequency for transmission, a signal bandwidth, a communications mode, and a set of time intervals in which transmission is allowed. The fundamental unit of transmission time is known as an interleaver block (IB). The communications mode consists of a modulation format and forward-error correction (FEC) code.

The system's frequency resources consist of several available bandwidths and center frequencies. Each epoch contains many IBs. Thus, it is possible for a large number of terminals to be given a time-frequency assignment in an epoch. Additionally, by varying the bandwidth, mode, and number of IBs assigned to a terminal, this MF-TDMA approach allows a wide range of data rates to be supported.

The process of making time-frequency slot assignments for each terminal consists of an allocation step and a packing step. The allocation step can depend on many factors, such as guaranteed rates, requested rates, their priority levels, and the amount of available resources. Once allocations in terms of mode and number of IBs are determined, a packing step assigns non-overlapping time-frequency slots to each terminal. If the assigned slots have large temporal gaps, packets transmitted may experience larger then desired delay and jitter.

Some user applications are sensitive to the amount of jitter in the received data, where jitter is defined to be the difference between the maximum (or 99% maximum, etc.) packet latency and the minimum packet latency. For example, a real-time IP voice application, such as voice-over-IP (VoIP), may require a small jitter to ensure a certain voice quality. Excess jitter could manifest itself by voice that is broken up from the end user's point of view, or it could even cause the application to drop packets arriving with large delays. Note that jitter is defined as a variation in latency, but we are concerned with reducing both jitter and latency. For example, if a delay and jitter sensitive application is provided with a large but constant latency, it would be desirable to decrease the latency if possible, even though there is no jitter.

The MF-TDMA system we consider introduces jitter into packet streams due to the TDMA nature of the allocation. For example, assume that a terminal is allocated three IBs which occur at time t_i, at time t_i+5, and at time t_i+20, respectively, and that the terminal is running an application that generates packets every 5 time units. Figure 1 provides an illustration of this IB assignment. If each IB can contain one voice packet, then the voice packets experience 15 time units of jitter due to the 15 time unit gap between the second and third IBs. If each IB can contain multiple voice packets, then the third IB could deliver the three arriving voice packets simultaneously. Thus, if hav-

[1] This work is sponsored by United States Air Force under Air Force contract #FA8721-05-C-0002. Opinions, interpretations, recommendations and conclusions are those of the author and are not necessarily endorsed by the United States Government.

ing bounded jitter is important to an application that a terminal is running, then the spacing of that terminal's IBs is one important consideration.

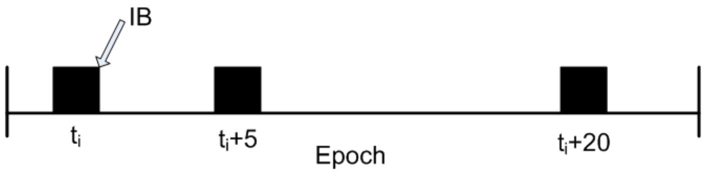


Figure 1: Example interleaver block allocation illustrating jitter due to assignments.

In this paper, we consider two allocation and packing algorithms that are designed to meet jitter constraints. The Jitter Reduction Algorithm (JRA), previously described in [1], statistically reduces jitter but does not strictly guarantee bounded jitter. The Jitter-Aware Algorithm (JAA) strictly guarantees bounded jitter but is more complex. We compare their performance using simulations.

FRAMEWORK

The MF-TDMA system we consider employs three primary time divisions. The system is similar to the one described in [1]. As mentioned previously, the epoch is the time unit over which terminal assignments are valid. Thus, the payload makes assignments for all of its terminals one epoch at a time. Modes, center frequencies, and communications bandwidths are not allowed to change during an epoch, but they can change from epoch to epoch.

The next time interval, the frame, is the fundamental unit of time employed by the terminal's and satellite's link layers. This means that data that is received during a frame cannot be passed to the decoder until the end of that frame. In our system, each epoch contains 32 frames, and epoch boundaries also coincide with frame boundaries.

Finally, an IB is the time unit employed by the FEC encoders and decoders. Specifically, each IB contains an integer number of FEC code words. No code word is received in its entirety until the final bit of the IB is fully received. In our system, each epoch contains 72 IBs. Since 72 is not an integer multiple of 32, IB boundaries do not necessarily coincide with frame boundaries. When an IB is fully contained in a single frame, the IB is called a non-spanning IB. Otherwise, the IB is known as a spanning IB. Figure 2 provides an illustration of the epoch, frame, and IB time divisions. Note that the four frames shown contain a total of six non-spanning IBs and three spanning IBs. An epoch of 32 frames contains eight such four-frame patterns.

The distinction between spanning and non-spanning IBs is an important one in our system. This is because each code word must be fully received to be decoded. By definition, all code words of a non-spanning IB are fully received by the end of the frame in which the IB is transmitted, so these code words are immediately passed to the decoder. On the other hand, a spanning IB takes at least two frames to be fully received, so none the code words it contains can be passed to the decoder until the end of the frame containing the final bits of the IB. Therefore, we wish to avoid, if possible, the use of spanning IBs for delay sensitive applications, since spanning IBs always cause additional transmission delay compared with non-spanning IBs.

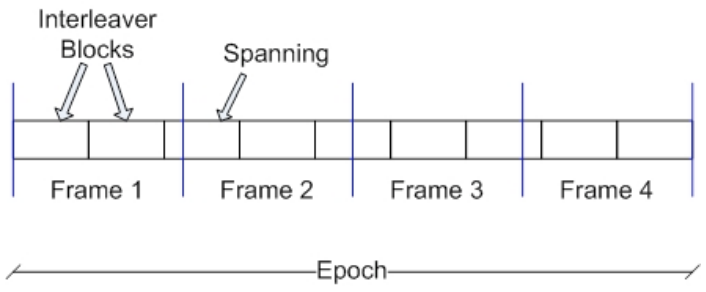


Figure 2: Epochs, frames, IBs, and spanning IBs.

The frame-based aspects of the system mean that even if a terminal were to be granted all of the non-spanning IBs of an epoch, code blocks and thus packets can only be formed at the end of each frame, implying that egress can only occur 32 times in each epoch (i.e., at the end of each frame of the epoch). Thus, our system always adds jitter to streams whose interarrival times are not multiples of the frame duration. If the stream's interarrival time is exactly a multiple of the frame duration and the assignment provides a sufficient rate, then there exists an IB assignment that adds no jitter to the stream.

JITTER REDUCTION ALGORITHM

The JRA is a relatively simple algorithm that statistically reduces jitter but does not strictly guarantee bounded jitter. With the JRA, first, resources are allocated to each user such that the Bandwidth Time Product (BTP) of the allocations does not exceed the total amount of system resources. Whenever possible, users are placed in narrower bandwidth modes requiring more IBs to meet the same rate demand. As more IBs are assigned, jitter is naturally reduced.

The next step is the packing of these IB allocations to each user into the system's available time-frequency resource. The packing algorithm is described in more detail in [1]. The basic concept is that larger bandwidth users are packed first. Of the users with the same bandwidth, packing is done in decreasing order of the number of IBs assigned. Each user is packed contiguously at the topmost slot (in frequency) available. If multiple slots at this frequency are available, the leftmost IB in time is selected. This algorithm has been shown to pack very efficiently but does not evenly space assignments within the epoch; rather, it intentionally bunches assignments together. Figure 3 shows the user assignments after packing and how an individual user's assignment is bunched together. Note that such bunched assignments actually maximize jitter.

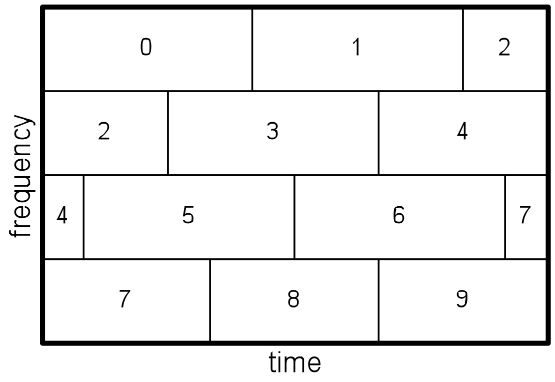


Figure 3: Standard Packing

To reduce jitter, a time-shuffling of the packing space is performed. Taking advantage of the fact that the initial packing step typically creates contiguous time assignments for each user, the shuffling step attempts to shuffle the assignments in time such that assignments that were close together are now far apart.

The shuffling is defined as

$$m = n * \beta \bmod T \tag{1}$$

where, m is the time index after shuffling, n is the time index before shuffling, β is the spreading factor, and T is the total amount of time per epoch measured in IBs.

To illustrate assume the packing shown in Figure 3 is composed of 12 IBs along the time axis. Then we can use (1) with $\beta = 5$ to yield the shuffled packing shown in Figure 4. In this example, assignments that were contiguous and bursty are now spaced at least 1 IB apart. Selection of a good spreading factor is very important in this approach.

For our system of interest with 72 IBs along the time axis, a spreading factor of 11 is used. This algorithm can guarantee that contiguous assignments can be evenly shuffled and thus jitter can be bounded. However, it is possible that assignments from the initial packing algorithm will not always be contiguous. In such cases, the shuffling step could take two of a user's noncontiguous IBs and shuffle them so they are back to back, which could increase jitter. This case is infrequent, but it cannot be guaranteed to be avoided. As a result, bounded jitter cannot be guaranteed. The probability of noncontiguous IBs being assigned is dependent on the distribution of user bandwidths and demand.

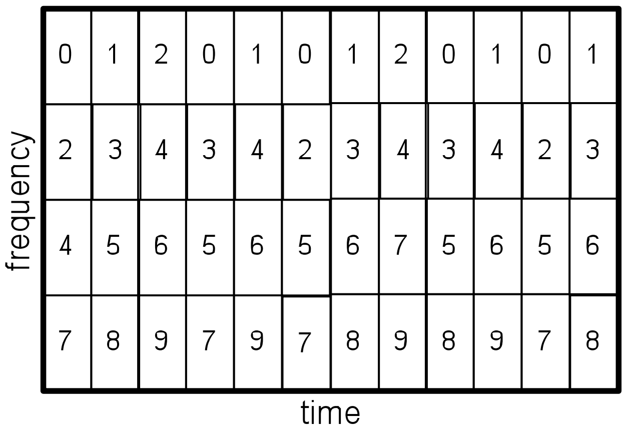


Figure 4: Shuffling of Standard Packing

JITTER-AWARE ALGORITHM

The second algorithm discussed in this paper is the Jitter-Aware Algorithm (JAA). The JAA strictly guarantees bounded jitter by assigning IBs in groups, referred to as IB groups or IBG. Within each IBG, IBs are uniformly spaced, thereby efficiently guaranteeing a certain minimum IB spacing and jitter. For example, taking one non-spanning IB from each frame forms a 1-frame IBG with 32 IBs. Assignment of one such 1-frame IBG guarantees IB spacings of one frame, and would allow a jitter equivalent to one frame to be achieved. Similarly, taking one non-spanning IB from every other frame forms a 2-frame IBG with 16 IBs. Other IBGs used in this algorithm are 4-frame IBG, 8-frame IBG, and 16-frame IBG, each with 8, 4, and 2 IBs, respectively. Spanning IBs are not used to form IBGs, and are assigned as single units of IBs.

Within each frequency band, there are a limited number of IBGs. As described in Figure 2, our system of interest contains a repeating pattern of 2,1,1,2 non-spanning IBs and 1,1,1,0 spanning IBs. Because some frames have only one non-spanning IB, there can be only one 1-frame IBG per frequency band. This one 1-frame IBG may also be split into two 2-frame IBGs or four 4-frame IBGs, and so on, or in combinations. In addition to these 32 non-spanning IBs, one from each frame, there are another 16 non-spanning IBs. These can be used to form two 4-frame

IBGs, four 8-frame IBGs, and so on, or in combinations. Finally, there are 24 spanning IBs. The number of each IBG available and how they can be exchanged is described in Figure 5. Generally, the 32 IBs that form a 1-frame IBG are considered the most valuable, since they can be used to form any type of IBG; the 24 spanning IBs are the least valuable.

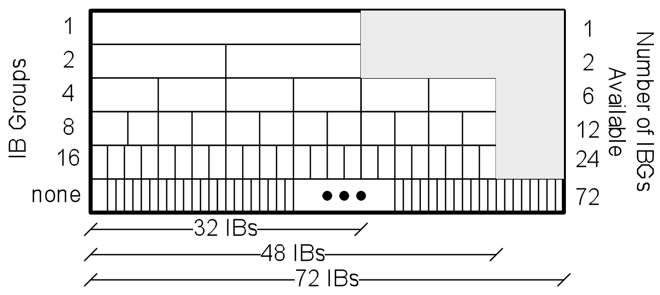


Figure 5: IB Group Relationships

Similar to the JRA, the JAA consists of an allocation step and a packing step. In the allocation step, the algorithm not only makes sure that the total BTP allocated does not exceed the total amount of system resources, but it also ensures that the allocation of IBGs at different jitter levels does not exceed the amount available. In particular, the algorithm ensures that the total amount of 1-frame and 2-frame IBGs allocated do not exceed the equivalent of 32 IBs per frequency band and that the total amount allocated for all IBGs of all jitter levels do not exceed the equivalent of 48 IBs per frequency band.

At the end of the allocation step, each terminal is allocated a number of different IBGs. For example, a terminal may be allocated one 2-frame IBG, two 4-frame IBGs, and four spanning IBs. In the event that a terminal's desired jitter cannot be met, the next-best assignment available is given. For example, if a terminal desires one 2-frame IBG, but there are not enough resources available, it may be given two 4-frame IBGs or even just 16 single IBs. The motivation is that when the desired jitter cannot be achieved, the allocation algorithm would at least try to meet the demanded rate and give the best jitter possible for that rate.

To efficiently utilize resources, excess allocations to meet a tighter jitter bound are salvaged by demands with looser jitter bounds. For example, assume a terminal requires an allocation of 80 kbps with 1-frame jitter to support a voice call and 300 kbps with no jitter requirement to support a data transfer, totaling 380 kbps. Also suppose each IB assigned supports 10 kbps, so one 1-frame IBG with 32 IBs can support 320kbps. This terminal would then be assigned one 1-frame IBG and just 6 single IBs to support both the voice and data transfer. The 320-80=240 kbps excess allocation with 1-frame jitter is salvaged by the data transfer demand.

The next step is to pack the IBG and IB allocations into the available time-frequency space. While the allocation step ensures that the total amount of allocation does not exceed the available amount, it is not guaranteed that all the allocations can be packed. Additional constraints that sometimes prevent complete packing include the terminal must be assigned the same IB in different frequency bands and a wideband assignment must occupy contiguous frequency bands.

To ensure that as much of the allocation as possible gets packed, the general philosophy of the packing algorithm is to pack the bigger, more constraint pieces first, in three stages. The algorithm first packs all of the largest bandwidth terminals, just as in JRA. Secondly, within one bandwidth, the algorithm first packs the largest IBGs with the tightest jitter bound, meaning all allocations of 1-frame IBGs of a particular bandwidth are packed first, and then the 2-frame IBGs, then the 4-frame IBGs, so on, and finally allocations of single IBs. Note that each terminal may have IBGs of different jitter levels assigned, so a terminal with a 1-frame IBG allocated only gets that allocation packed first, and the rest of its allocation has to wait until all other 1-frame IBG allocations to other terminals are packed. Finally, within one bandwidth and jitter level, allocations of more IBGs or IBs are packed first, just as in the JRA. When packing single IBs, the 24 spanning IBs are used first, since they can not be used to form IBGs. When packing IBGs or IBs, the 32 IBs that form a 1-frame IBG are used last, since packing any looser jitter IBGs there would prevent a 1-frame IBG from being packed. This corresponds to using IBs from right to left in Figure 5. In the event that an allocation of an IBG cannot be packed, it is broken up minimally so that it can be packed. For example, if a 2-frame IBG can not be packed, it is broken into two 4-frame IBGs or even just 16 single IBs.

When a terminal is allocated multiple IBGs or IBs, the algorithm also attempts to make the assignments well spaced, similar to the shuffling in JRA as shown in Figure 4. While this does not affect the 1-frame and 2-frame IBGs, it is significant for the smaller IBGs. For example, if one terminal is allocated two 16-frame IBGs, each with two IBs, the algorithm attempts to make the two pairs of IBs well spaced to resemble an 8-frame IBG. This is done by a simple table-lookup mapping such that the IBGs with neighboring indexes map to groups of IBs that are relatively well spaced.

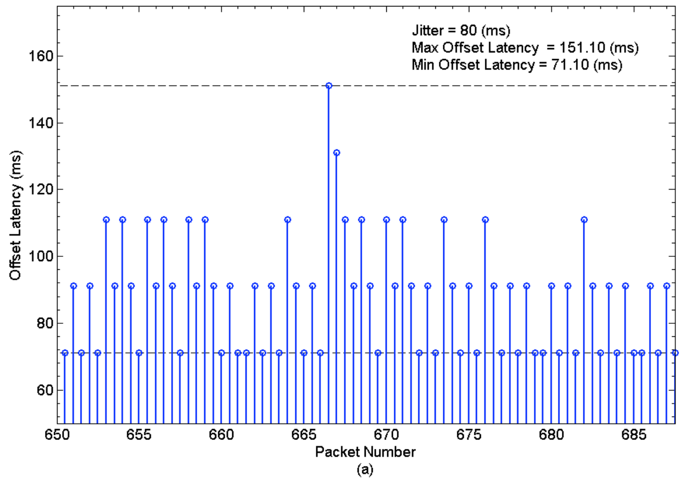
An additional feature of the JAA is that it attempts to limit epoch-to-epoch changes in a user's IB locations if the user's mode has not changed. To illustrate a potential problem with epoch-to-epoch changes, consider a user who is assigned a single IB in each of two consecutive epochs. In the first epoch, the IB is contained in the first frame after packing. In the next epoch, however, the IB is contained in the last frame after packing. The spacing between these two IBs is nearly two epochs long. A better packing would be to place the IBs in the same location in each epoch, which limits the spacing to approximately one epoch long and causes less delay and jitter. To avoid unfavorable epoch-to-epoch spacing, the algorithm assigns the same IB locations when possible. Finally, when a user is assigned additional or new IBs, the packing strategy described earlier is used to pack the IBs.

Given the complexity of assigning and packing with bounded jitter, this algorithm is significantly more complex than the JRA shuffling technique.

RESULTS

The numerical results presented here were obtained via OPNET simulation of a system with approximately 100 users, each with dynamic traffic demands, including bounded-jitter applications and non-bounded-jitter applications. Each epoch contains 32 frames and 72 IBs. We focus on one of the users who, in addition to its dynamic traffic, also has a bounded-jitter application whose start and end times have been scripted. This allows each of the algorithms to be evaluated under controlled traffic conditions at a common reference point.

Results for the jitter-reducing and jitter-aware algorithms are presented in Figure 6 and Figure 7. The packet latency for the bounded-jitter application is presented as a function of the packet number. Note that the latency values in Figures 6 and 7 are offset by a fixed value due to unmodeled effects (e.g., propagation delays). In Figure 6 we show the results for a scenario in which the traffic for the bounded-jitter application arrives at a rate of one 320-byte packet every 20 milliseconds (ms). Figure 6(a) includes results for the JRA. For this algorithm, the packet-to-packet jitter is typically less than 40ms, but at times it reaches 60 ms and even 80ms. The average number of non-spanning and spanning IBs assigned per epoch is 14.38 and 7.15, respectively. The maximum IB spacing is 100 ms. The results for the JAA for the same traffic scenario are shown in Figure 6 (b). Compared with the results in Figure 6 (a) for the JRA, the JAA provides much better performance. The maximum jitter is 20 ms. This is due not only to the exclusive use of non-spanning IBs, but the IB spacing is also allocated intelligently by this algorithm. Specifically, the average number of IBs (all non-spanning) assigned by the jitter-aware algorithm is 24. The maximum IB spacing is 40 ms, which implies a much more uniform spacing than the spacing produced by the JRA.



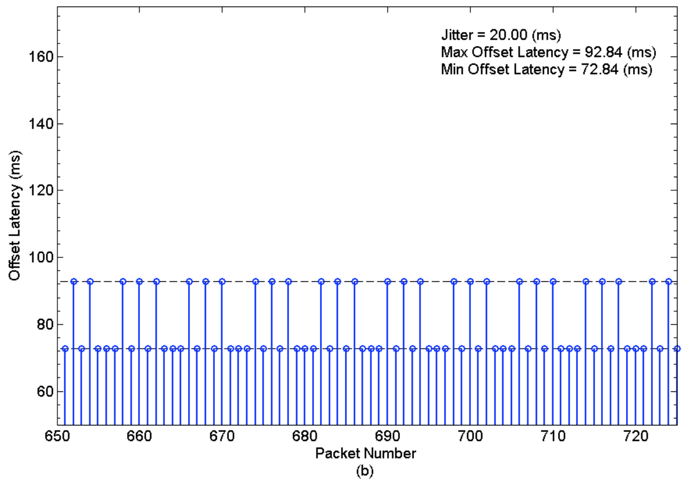


Figure 6: Experimental simulation results comparing the jitter performances of the JRA (a) and JAA (b) algorithms

In Figure 7 we show the results for the traffic scenario where the packets arrive at a rate of one 640-byte packet every 40 ms. Note that the maximum jitter is typically 40 ms for the JRA (Figure 7(a)), but the jitter is zero for the JAA (Figure 7(b)). The average number of non-spanning and spanning IBs for the JRA are 14.27 and 7.22, respectively. The maximum IB spacing is 40 ms. For the JAA, the average number of non-spanning IBs, average number of spanning IBs, and maximum IB spacing are the same as in Figure 6.

The reduced jitter, however, is obtained at the expense of system resources. Recall that for the JAA, the best jitter bound achieved with a 1-frame IBG can only be given to a single terminal per frequency band. In a typical operation scenario, there are many more terminals than frequency bands, which implies that only a few terminals are able to achieve the best jitter bound. In fact, resource overallocations for a few terminals may be so large that other terminals are left with insufficient resources to satisfy their required rate, let alone jitter. It is important to choose a reasonable jitter bound to balance jitter performance and system efficiency.

CONCLUSION

Two dynamic resource allocation techniques that also attempt to reduce or bound jitter have been described. The jitter-reducing algorithm is a simple technique that provides good TDMA spacing for interleaver blocks, but this algorithm cannot guarantee that jitter bounds will be met in all cases of interest. The jitter-aware algorithm is a much more complex algorithm, but it can provide bounded-jitter guarantees as long as sufficient resources are available.

REFERENCES

1. N. Yazdani, "Multi-Frequency Time-Division Multiple-Access (MF-TDMA) Resource Packing," *Proceedings of the 2008 IEEE Military Communications Conference (MILCOM)*, November 2008.

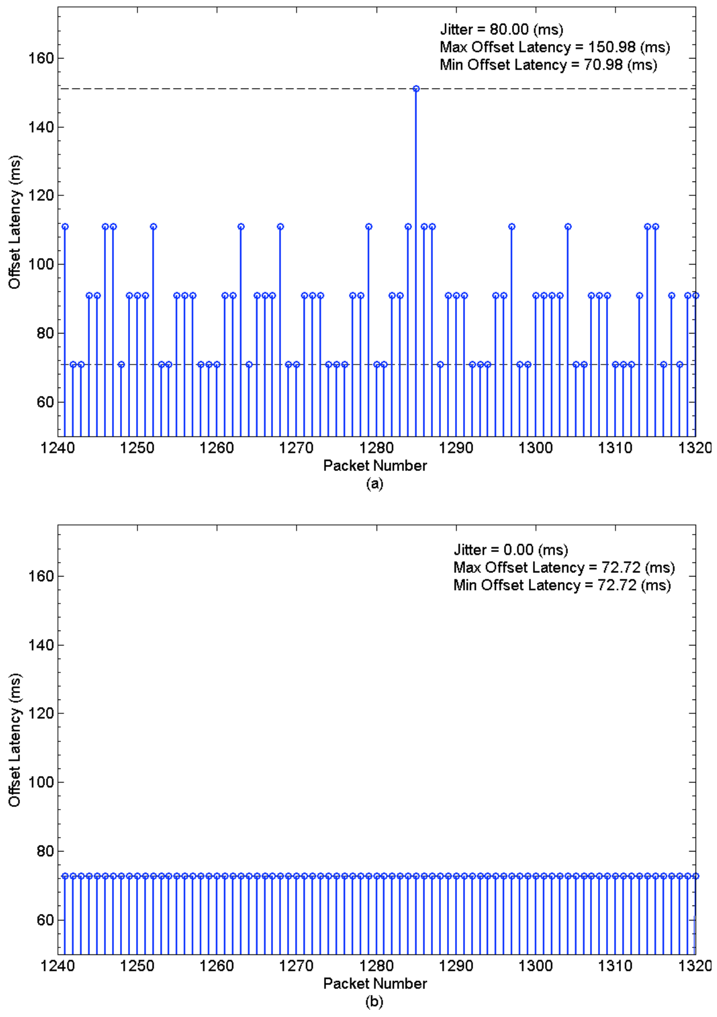


Figure 7: Experimental simulation results comparing the jitter performances of the JRA (a) and JAA (b) algorithms in which voice packets arrive at rate of one 640-byte packet every 40 milliseconds.