Abstract—Measuring the physical layer performance of a wireless radio transmission is straightforward: Firstly, a transmitter generates the signal-samples to be transmitted. Secondly, these samples are broadcasted in real-time and captured by the receiver. Thirdly, the received signal is evaluated and the desired figure of merit is calculated. If only isolated blocks of data are transmitted and the received signal is evaluated off-line to simplify implementation, the receiver has to know when to actually acquire the data. The situation becomes more complicated when several transmitters and receivers have to be synchronized in time.

In this paper we will present a method to synchronize an arbitrary number of transmitters and receivers that are swift (e.g. 1 ms, the latency of an already exiting LAN infrastructure) as well as accurate (e.g. ±30 ns, the relative accuracy of GPS).

I. MOTIVATION

In currently employed wireless radio transmission systems, receivers usually capture the input signal on a continuous basis [1]. In a next step, more or less complex digital signal processing algorithms continuously estimate the most likely start of data blocks to then decode the received information. This important step of finding where the blocks of data actually are is usually referred to as “timing synchronization”.

In contrast, a well established way to infer the performance of complex wireless communications systems is to create the blocks to be transmitted off-line in a high-level programming language, transmit and receive only single blocks of data, and then evaluate the received blocks off-line [2]. Such “testbed measurements” avoid the cumbersome implementation of algorithms in hardware while still allowing for transmissions over real-world channels, therefore capturing real-world effects.

When now only a single block of data is to be transmitted, and the receiver is only capturing data for a limited amount of time (that is, in the optimum, approximately the time required for the transmission), the receiver has to know somehow when to actually receive the data. If data processing is carried out only off-line after the acquisition has finished, the received data itself cannot be used for this purpose and, therefore, an external triggering mechanism has to be employed. Such triggering mechanism does not necessarily have to be very accurate if more data is captured than actually required. In this case, accurate time synchronization can be carried out later off-line in software. On the other hand, accurate hardware-triggering and therefore hardware-synchronization may simplify the work of the researchers as either the off-line time synchronization can be omitted or the not-omitted timing synchronization can be compared to the perfect reference of a hardware trigger.

Things change when several time-synchronous transmitters are used in an experiment as software-aided synchronization techniques cannot be employed anymore. This may be for example the case when measuring the performance of interference alignment\(^1\) techniques [3].

\(^1\)The key idea of interference alignment is to shape multiple transmit signals in such a way that they cast “shadows” on other users, while the user(s) of interest can still decode them.

The goal is therefore to design a “distributed piece of hardware” that allows for precise synchronization of several transmitters and receivers at a low cost while still allowing for the flexibility of an off-line measurement approach. Specifically, it should be possible to transmit blocks of data immediately upon request, and not only at full seconds as it could be easily realized by using a PPS\(^2\) signal from a GPS reference.

Indoors, or when the sites of the measurement system are not located far from each other, the probably easiest implementation of such a distributed timing synchronization system is a 50 ohm coaxial cable to spread a common trigger signal. If however, as shown for example in Figure 2, the transmit sites are located on different

\(^2\)A Pulse Per Second (PPS) is an electrical signal that is high for a certain period starting at exactly each full second. It does, therefore, not specify the time but only the start of each second.
buildings in a city, laying dedicated cables becomes non-feasible, calling for a new, innovative approach.

II. THE CONCEPT

The simplest way to exchange information between different sites is to use/extend a typically already existing LAN/WAN\(^3\) infrastructure and/or maybe also set up dedicated wireless LAN bridges to connect different LAN segments. Unfortunately, if not specially designed for this purpose\(^4\), LAN connections suffer from a typically non-constant delay. Therefore, they can only be used for information exchange between the geographically separated sites of a testbed but not for supplying the trigger instant itself.

Precise timing (±30 ns relative difference\(^5\)) at very different sites can be easily obtained using the PPS signal provided by commercial-grade GPS devices. However, these devices do not allow for additional information exchange between the different transmitter and receiver sites.

The basic idea now is to use a GPS at each site to acquire the current time precisely, and a local area network connection between all sites to handshake at what time to trigger the transmission. Figure 3 shows the proposed set-up, employing a dedicated timing synchronization unit at each testbed site:

Fig. 3. Block diagram of the set-up proposed. Parts operating in real-time are shown in gray. We actually employ three transmitters and one receiver. The methodology proposed works with any number of transmitters and receiver sites.

A. Initial Preparation

Prior to a measurement (and not during a measurement), the following steps have to be carried out:

1. All GPS units acquire a satellite fix. (Duration: sometimes up to several minutes but not relevant for the measurement)
2. The master PC (typically a receiver) tells its own sync-unit via a Software execution, origin is applied.

\(^3\)A Wide Area Network (WAN) is, in contrast to a Local Area Network (LAN), covering a much larger geographical area. For simplicity, we will refer to both, WANs and LANs, as LANs throughout this paper.

\(^4\)Defined in IEEE 1588 [4], the Precision Time Protocol (PTP) allows for precise timing synchronization using “special” network cards and switches. While at laboratory conditions a timing performance close to GPS devices can be achieved, already existing networks do not allow for such performance [5].

\(^5\)We measured this value using commercial-of-the-shelf equipment [6, Figure 3.14.]. Data sheets of GPS devices typically report the standard deviation of this difference, which is obviously lower than its maximum values.

\(^6\)A User Datagram Protocol (UDP) datagram is a basic transfer unit of the packet-switched LAN. In contrast to Transmission Control Protocol (TCP) packet transmission, UDP datagram transmission does not provide reliability, ordering, and data integrity. On the other hand, avoiding the therefore required overhead, it is more suited for real-time applications.

\(^7\)FIFO...In our case a dual-port memory is set up as a First In First Out buffer (FIFO buffer) to first store the off-line generated samples and then forward these samples to the DAC in real-time. The buffer is required as the software is not capable of generating the data samples fast enough.

Fig. 4. UDP datagrams on the LAN (arrows) plotted over time (Y-direction).

(a) The master PC (typically a receiver) tells all other PCs that the transmission of a specific data block shall be performed. (UDP multicast, duration: maximum LAN latency)

(b) - In all transmitters, the specific data block to be transmitted is created off-line in the block named software (see Figure 3). In the form of samples, this data is then stored in the block named FIFO\(^3\) to be transmitted exactly when the external trigger originated in the sync-unit is applied. (Software execution, start: when UDP multicast is received, duration: unknown)

- All receivers get ready for reception.

(c) - Each transmitter tells its own sync-unit that a transmission shall be triggered in \(t_d\) milliseconds. The time \(t_d\) is to be determined...In our case a dual-port memory is set up as a First In First Out buffer (FIFO buffer) to first store the off-line generated samples and then forward these samples to the DAC in real-time. The buffer is required as the software is not capable of generating the data samples fast enough.
by the user beforehand or to be optimized adaptively during operation.

- The receivers are assumed to be ready and do not necessarily notify their sync-unit. In case this assumption is not valid, an error is reported in Step (f). (UDP unicast, start: when samples are in FIFO, duration: single switch latency as PC and corresponding sync-unit are not geographically separated)

(d) Each sync-unit receiving a trigger command looks up its own internal clock and multicasts this time plus \( t_L \), as the earliest possible trigger time to all sync-units. (UDP multicast, duration: maximum LAN latency)

(e) All sync-units wait until they receive from every transmitter (the number of transmitters is known beforehand) the desired trigger time. Then they trigger at the latest time received. This time is exactly the same for all units and all units must be able to trigger at this time if \( t_L \) is higher than the maximum UDP datagram delay \( t_{UDP} \) observed (see Figure 4). (Wait until trigger which happens exactly \( t_L \) after the slowest PC has requested a trigger)

(f) In case the datagram with the message to trigger has arrived after the triggering time contained in it (this happens if \( t_L \) was set too low), the units report an error and the transmission has to be repeated, otherwise the successful triggering is reported back.

By increasing \( t_L \), the rate of success can be increased while at the same time the duration of the triggering procedure is incremented. The optimal \( t_L \) (that is the unknown non-deterministic \( t_{UDP} \)) can be set adaptively as a tradeoff between the time required for repeating a transmission and the time \( t_L \) that is lost for carrying out the triggering procedure.

(g) The data received is evaluated. Note that the evaluation already starts when the data is received. It is not necessary to wait for the “successful triggering” reported in Step (f) to start the evaluation. In case an error is reported instead of a “successful triggering,” the evaluation is aborted.

If an UDP datagram is lost during the above procedure, neither a successful triggering nor an error is reported back. In this case, a timeout occurs at the receiver, all sync-units are reset, and the whole procedure is repeated.

C. GPS (+ Rubidium) As Time Reference

In order to trigger at the same time instant, all sync-units need very stable internal clocks. Low cost (commercially available) GPS modules that provide a PPS output next to a stabilized 10 MHz oscillator output (for example [7, 8]) may be already sufficient to provide precise timing information in most measurement set-ups.

If, however, a higher short time phase and frequency stability is required, a rubidium standard may be the device of choice. Commercially available frequency standards (for example [9]) provide long-term (via a GPS) and short-term (via the rubidium) stabilized 10 MHz as well as PPS outputs (see Figure 5). The price to be paid for a time stability factor of \( \pm 5 \times 10^{-11} \) is in the magnitude of 3 kEUR per sync-unit to stabilize.

D. Precise DAC Triggering and Frequency Synchronization

Although the trigger pulse generated with the set-up proposed in Figure 5 is very precise, the output of the DAC itself will experience a timing uncertainty of one DAC clock cycle\(^6\) as the DAC will quantize the trigger pulse to the “next” edge of the DAC clock. To avoid this uncertainty, the DAC has to operate on the same clock as the sync-unit.

\(^6\)Note that with some DACs this uncertainty is even higher.

- Fig. 5. Enhanced by a rubidium frequency standard at each site (only one transmitter is shown in the figure) the proposed set-up is able to trigger with a short-time precision factor of approximately \( 5 \times 10^{-11} \). The long term accuracy of approximately \( \pm 30 \) ns is defined by the GPS used.

Figure 6 shows such a set-up where DAC and sync-unit are operated on the same externally generated clock, allowing for a trigger pulse that has a constant timing relation to the DAC clock. If this externally generated clock (see the block labeled “oscillator” in Figure 6) is stabilized by the 10 MHz output of the rubidium, hardware frequency synchronization of all units in a testbed can be achieved as a side effect. If desired, the local oscillator of the up-conversion can also be locked to the rubidium reference. The resulting set-up can then even be used for channel sounding.

- Fig. 6. If the sync-unit and the DAC are operated on the same external clock, precise triggering is possible. If all oscillators are locked to the rubidium reference and the scenario is static, consecutive transmissions will experience the same channel. Note that we define the channel from the DAC to the ADC, not from TX antennas to RX antennas.

III. DISCUSSION OF THE SET-UP PROPOSED

A. Applicability

The set-up proposed allows for timing synchronization of several transmitters and receivers over an already existing LAN connection. With less triggering speed (equal to the network latency), synchronization can be even achieved over low-cost 3G/4G wireless modems (e.g. 50 ms per trigger in HSDPA).

Please note that the units do not need to be connected directly by a cable in order to synchronize precisely. A LAN connection, a GPS, and a rubidium frequency standard that is at best never turned off\(^9\) is sufficient.

B. Validation and Verification

The set-up can be easily validated\(^{10}\) by displaying the trigger pulses of all units on a single oscilloscope.

\(^9\)We power each rubidium frequency standard via a Uninterruptible Power Supply (UPS) so that we never have to turn them off.

\(^{10}\)Validation: “Has the thing been built right?” Verification: “Has the right thing been built?”
As we are interested in testing wireless communication systems, a way to verify the proposed set-up is to transmit a signal from several transmitters, acquire it, and then estimate (1) the beginning of the received block as well as (2) the channel. When repeating the transmission, these two estimates should not change significantly in a static environment.

C. Repeatability

If repeating an experiment requires a transmission to happen at a specific time the proposed set-up may not be sufficient. The reason is that firstly, in Step (c) in Figure 4, a packet has to be sent to the sync-unit resulting in an unknown delay next to the fixed time \( t_s \) that cannot be avoided. Secondly, it is very hard to time UDP datagrams in software more accurately than in a microseconds [10]. We therefore implemented an external hardware input into the sync-unit.

D. Achievable Triggering Speed

When transmitting blocks of data, the latency introduced by the triggering is the sum of:

1) the time required to tell the sync-unit to trigger (Step c) in Figure 4), that is, the delay of a network switch,
2) the time required by the sync-unit to receive and transmit UDP packets as well as to read the internal clock (the vertical lines of Steps d) and e) in Figure 4), that can be made negligible by employing high-speed hardware, and
3) the estimated maximum time required to broadcast the trigger time over the network (Step d) in Figure 4).

Note that the time required to receive the success report (step e) in Figure 4) is not included in the sum, as the received data can already be evaluated (Step f) in Figure 4) while waiting for the success report.

Therefore, if only triggering, triggering speeds of more than one trigger per millisecond can be easily achieved in typical LANs.

However, in a real measurement, the transmission not only has to be triggered but data also has to be generated, transmitted, and received. If employing off-line processing of data, the triggering itself will only account for a negligible part of the overall measurement time (far less than 10%). It is therefore not necessary to optimize the speed of the triggering procedure anymore.

IV. Our Hardware Implementation of the Sync-Unit

The synchronization unit has been implemented in hardware (see Figure 7) and tested in various local area network environments. For example, if “generating the TX samples and loading them into the FIFOs” (step b in Figure 4) takes 4 ms and the data is transmitted while the next block is already loaded, a rate of 233 transmitted data blocks per second has been reached in a local area network. A detailed analysis can be found in [11].

V. Conclusions

In this paper we presented a method to synchronously trigger hardware at different locations using a GPS and a dedicated sync-unit at each site, as well as an already existing LAN infrastructure. We showed that the precision of the overall set-up can be further increased by employing rubidium frequency standards and external local oscillators.

Fig. 7. Sync-unit built. The unit incorporates a host of additional input/output connections in order to control other hardware such as step attenuators, linear guides, rotation units, and radio frequency switches.

VI. Acknowledgements

This work has been funded by the Christian Doppler Laboratory for Wireless Technologies for Sustainable Mobility, the A1 Telekom Austria AG, and the KATHREIN-Werke KG. It has also been supported by the Xunta de Galicia, the Ministerio de Ciencia e Innovacio n de Spain, and the FEDER funds of the European Union under the grants 10ITC003CT, IPT-020000-2010-35, TEC2010-19545-C04-01, and CSD2008-00010. The financial support by the Federal Ministry of Economy, Family and Youth and the National Foundation for Research, Technology and Development is gratefully acknowledged.

REFERENCES

http://publik.tuwien.ac.at/files/PubDat_174574.pdf
http://www.trimble.com/timing/thunderbolt-e.aspx
http://www.accubeat.co.il/
http://www.thinksrs.com/products/FS725.htm
http://www.endace.com
http://publik.tuwien.ac.at/files/PubDat_XXXX.pdf