

Synchronization of Single-Frequency Simulcast Networks Using Network Time Protocol

Stefano Bregni*, *Senior Member, IEEE*, Luciano Lacavalla*,
Bruno Propersi*, Francesco Residori**, *Member, IEEE*

* Politecnico di Milano, Dept. of Electronics and Information, Piazza Leonardo Da Vinci 32, 20133 Milano, ITALY

Tel.: +39-02-2399.3503 – Fax: +39-02-2399.3413 – E-mail: bregni@elet.polimi.it

** SELEX Communications Prod-El S.p.A., Via Palmanova 185, 20132 Milano, ITALY

Tel.: +39-02-25928.229 – Fax: +39-02-26300802 – E-mail: fresidori@prod-el.com

Abstract — Single-frequency simulcast networks use two or more Radio Base Stations (RBS) to transmit simultaneously the same signal on the same radio channel over the service area. To ensure correct operation and good performance, simulcast networks need accurate time and frequency synchronization. In this work, we designed a system for synchronizing RBS clocks by way of Network Time Protocol (NTP) packets. The goal is to allow simulcast operation when external sources of synchronization, such as GPS, are not available or when the operator prefers to distribute timing alternatively, in particular by IP network facilities used for data transport. Time stamps are disseminated by a master RBS and used in slave stations to steer local clocks. The mechanism proposed was modelled, simulated and implemented in an experimental prototype. Simulation and experimental measurement results meet requirements and demonstrate the practical feasibility of this approach for commercial application.

Index Terms — Digital radio, land mobile radio cellular systems, radio broadcasting, synchronization, time dissemination.

I. INTRODUCTION

Large coverage and spectral efficiency are significant characteristics in Private Mobile Radio (PMR) and Public Access Mobile Radio (PAMR) applications, in particular security and public utilities. A solution is given by *simulcast networks*, in which radio transmitters broadcast the same analog or digital FM signal [1]–[5]. Examples of digital systems for PMR are the Trans-European Trunk Radio (TETRA) [6], Digital Mobile Radio (DMR) [7], MPT 1327 signalling systems (audio sub carrier FSK + FM) [8], paging systems (POCSAG, ERMES), Personal Communication Systems (PCS).

In this paper, the interest is on FM analogue modulation (with channel spacing 12.5 kHz / 25 kHz) and on DMR digital modulation [7]. This is an ETSI standard for digital voice and data services, based on 4-FSK to achieve the bit rate 9600 bit/s with 12.5 kHz channel spacing and two time slot TDMA.

As shown in Fig. 1, single-frequency simulcast networks use two or more Radio Base Stations (RBS) to transmit simultaneously the same signal on the same radio channel over the service area. Downlink, a master station (RBS #1 in Fig. 1) covers its area and uses other slave stations (RBS #2, 3, 4 and 5) to improve the coverage. Uplink, the master station selects by majority voting the best received signal from different radio stations.

The links between stations may be UHF radio, cable (4 wires), digital circuits across Plesiochronous Digital Hierarchy

(PDH), Synchronous Digital Hierarchy (SDH) or switched Integrated Services Digital Network (ISDN) systems. A further possibility is connecting stations across an IP network.

In simulcast systems, signals with comparable levels, received from two or more RBSs in an overlapping area, cause quality degradation (i.e., higher Bit Error Rate, BER, in digital signals or worse signal-to-noise ratio in analogue signals). Moreover, frequency offset between RBSs causes both fades and clicks [9]. Reduction of amplitude imbalance increases the click amplitude.

In digital frequency modulation, a discriminator-based receiver exhibits a BER floor in a multipath fading and/or simulcast environment. This floor increases with speed of movement, frequency offset of transmitters and decrease of amplitude imbalance of the simulcast signal [10].

The delay difference among broadcasted signals, due to different transmission line lengths between stations, severely degrades system performance. Thus, adequate delay equalization among RBSs is required [11].

Simulcast networks are specified by ITU-R Rec. M.1077 [12]. ITU-R also specifies minimal synchronization operative conditions for acceptable system performance. Frequency synchronization requirements, for correct operation of simulcast networks, are specified in terms of maximum RF frequency offset between RBSs:

- 2 Hz in the VHF band (equivalent to $1.25 \cdot 10^{-8}$ fractional frequency offset at 160 MHz);
- 1 Hz in the UHF band ($2 \cdot 10^{-9}$ at 450 MHz).

The maximum offset 6 Hz should be never exceeded (equivalent to $3.75 \cdot 10^{-8}$ at 160 MHz and to $1.33 \cdot 10^{-8}$ at 450 MHz). Moreover, time synchronization requirements are:

- 5 μ s for analog signals;
- $T_s/10$ for digital signals;

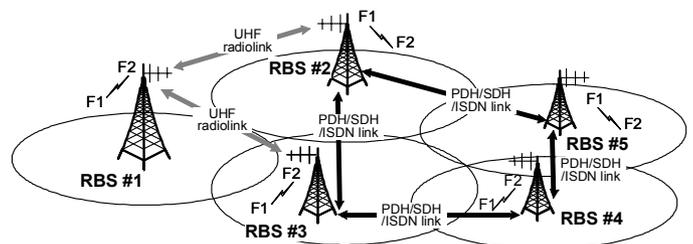


Fig. 1: Example of simulcast network.

where T_s is the symbol duration. These limits were found by long term simulcast trials to optimize system performance.

Synchronization of RBSs, in order to compensate different link delays, may be easily achieved by Global Positioning System (GPS) or similar systems. In spite of this, there is much interest among manufacturers for engineering alternative mechanisms of time distribution. In actual fact, many network operators outside America do not accept to deploy network systems that are depending on GPS availability for optimal performance. Main grounds are unwillingness to depend on decisions of a foreign administration and the fact that the GPS signal may be not available continuously and in all sites (e.g., in tunnels or close to TV/radio broadcasting antennas). Hence, the interest for designing alternative mechanisms for time distribution, in particular across data links connecting RBSs.

Over PDH/SDH links or circuit-switched networks, the transmission delay varies negligibly with time (mainly due to wander [13]) and thus can be easily measured and compensated. Network synchronization is more difficult when RBSs are connected over a packet-switched network (e.g., over IP). In this case, the transport delay varies significantly with time, depending on network congestion, the number of hops, etc.

The goal of this work is to study a mechanism to steer clocks of simulcast stations, by distributing the master station clock reference to all others, when RBSs are linked through an IP geographic or local network.

Network Time Protocol (NTP) is the standard protocol for time dissemination over the Internet. Developed by D. L. Mills [14]–[18], it allows to distribute time information from the master clock by way of IP packets. NTP packets are sent by the master under client request. Each NTP packet carries time-stamps, used to estimate the time difference between master and client clocks. The NTP server clock provides an absolute reference for the NTP clients. NTP performance in terms of timing accuracy achievable is astonishingly good, considering the jitter that affects IP packets forwarded through the Internet. This information can be used to correct the time offset between RBSs and to synchronize them in time and frequency.

Therefore, at Prod-El, we designed a NTP-based synchronization system of Simulcast RBSs connected via an IP network. This system was modelled, simulated and implemented in an experimental prototype. Simulation and experimental results meet requirements and demonstrate the practical feasibility of this approach for application in commercial systems. Experimentation on the prototype, on the other hand, was also useful to reveal implementation pitfalls.

This paper is organized as follows. Sec. II outlines the system architecture and model. Sec. III presents some simulation results. Sec. IV illustrates the prototype, provides a few details about hardware and firmware implementation and shows measurement results. Finally, Sec. V draws some conclusions.

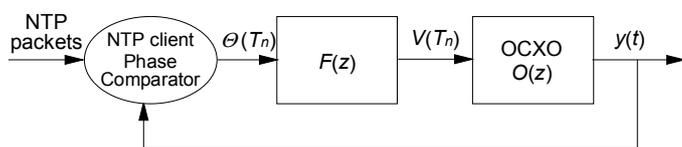


Fig. 2: PLL model of RBS synchronization by NTP.

II. SYSTEM ARCHITECTURE AND MODEL

Each RBS has an internal software clock [19], with the reference frequency obtained from an Oven Controlled Crystal Oscillator (OCXO), with fractional frequency accuracy on the order of 10^{-6} within a wide temperature working range from -20°C to $+70^{\circ}\text{C}$.

Network synchronization is achieved according to the well-known Hierarchical Master Slave strategy. This strategy is based on the distribution of the timing reference from one clock (master clock) to all the other clocks of the network (slave clocks), directly or indirectly, according to a star (two-level) or tree (multi-level) topology.

In our architecture, the master station acts as NTP server (reference) and slave stations as NTP clients. Periodically, at a given polling time, clients send NTP request packets to the server and wait for response packets. From time stamps read in response packets, clients estimate the round trip time and thus the master clock time. From this information, they can estimate their clock delay and frequency offset.

Therefore, at approximately each polling time (the response packet arrival time varies, depending on the network traffic and the number of hops), the client can estimate its time offset and drift, by comparing the server clock to the local one. To adjust the software clock, the client actually tunes the frequency of the OCXO, by acting on its control voltage.

This mechanism is basically a Phase-Locked Loop (PLL) [13], in which the NTP client has the role of phase comparator. This is a discrete-time system, timed by the polling period T (phase error is computed when the NTP response packet arrives). We decided to design a 3rd-order, type-2 PLL. A type-2 PLL has an integrator in the loop filter and is able to track input frequency steps with null residual phase error.

The choice of the polling period T (system sampling time) can be critical. A long polling period slows down synchronization convergence, while a short polling period can affect loop stability. Moreover, a short polling period could improve phase tracking precision, but the IP network might be also excessively loaded by NTP traffic. We believe that the correct choice depends on the specific case. In our prototype system, the polling period can be set by software control.

A model of this RBS synchronization system is shown in Fig. 2. Transfer functions are given in the z domain, as we deal with a discrete-time system. At each sample of clock offset $\theta(T_n)$, the filter $F(z)$ returns a sample $V(T_n)$ to tune the frequency of the RBS OCXO, which outputs the periodic timing signal $y(t)$. The type-2 PLL filter transfer function is given by:

$$F(z) = G_1 \frac{z}{(z-1)} \frac{(z-z_z)}{(z-z_p)} \quad (1)$$

where z_z and z_p are the zero and pole values, respectively, and G_1 is the filter gain. Parameters are designed to ensure system stability. Thus, the PLL open-loop transfer function is:

$$L(z) = F(z) \cdot O(z) = G_2 \frac{z^2}{(z-1)^2} \frac{(z-z_z)}{(z-z_p)} \quad (2).$$

As far as convergence speed is concerned, a wideband $L(z)$ is faster than a narrowband one, but it tracks input asymptoti-

cally with less precision, after the transient is completed. To speed-up convergence, the so-called *Gear Shifting* method is used. At the beginning, a fast transfer function is used, in order to achieve convergence rapidly. Subsequently, after coarse synchronization has been acquired, the system switches to a slow transfer function to attain the precision required.

The filter is switched when the phase error, measured by the NTP client, becomes smaller than a fixed threshold. Should the input phase error become unexpectedly large, e.g. due to IP network congestion, the filter switches back to the wideband one, in order to speed-up reacquisition.

III. SIMULATION RESULTS

The system described in the previous section has been simulated in Matlab Simulink[®], for feasibility and performance evaluation before designing a hardware prototype.

The simulation model is shown in Fig. 3. Every block is sampled with sample time T , the polling period of our system. The *RBS NTP server* block is the base station with its OCXO reference clock that acts as NTP server. The *RBS NTP client* receives NTP packets, processes them and calculates the time offset $offset(T_n)$ respect to the local clock (*clock counter*).

The value computed by the NTP client is passed to the filter $F(z)$, which outputs $Vtune(T_n)$, which controls the OCXO frequency. The OCXO block outputs the periodic signal

$$y(t) = A_c \cos \left[2\pi f_c t + K_{OCXO} \int_0^t Vtune(t) dt + \varphi \right] \quad (3)$$

where A_c is the output amplitude, f_c is OCXO free running frequency, K_{OCXO} is the OCXO control sensitivity and φ is the initial phase. The OCXO block output is used to increase the local clock counter, whose output is looped into the NTP block. The polling block is used to sample the NTP phase detector at each polling time T .

The noise block adds noise to the input signal given by the NTP server. White Gaussian noise was used to evaluate the filter and to calibrate its parameters. Note that the NTP algorithm, when calculating the time offset between client and server, is affected by various noise contributions, like timestamps detection, measure errors and variable network delay.

Examples of simulation results are shown in Fig. 4, where $offset(T_n)$ and $Vtune(T_n)$ are plotted when the NTP reference time is perturbed by additive white Gaussian noise with deviation $\sigma_n = 20 \mu\text{s}$ and by an initial frequency step $\Delta f = 10^{-6}$ (client slower than server). Initial time offset was set to zero. In the second graph, moreover, the plot of $Vtune(T_n)$ exhibits evidence of the mentioned Gear Shifting method, after nearly 2800 seconds. In the first phase, the wideband filter achieves fast but coarse convergence, then the system switches to the slow filter, which improves acquisition accuracy.

Table 1: Time needed to converge within ± 1 Hz of frequency error.

Noise standard deviation	Fast filter -3dB bandwidth	Slow filter -3dB bandwidth	Time of convergence
$\sigma_n = 20 \mu\text{s}$	$4.4 \cdot 10^{-1} \text{ rad/s}$	$1.4 \cdot 10^{-1} \text{ rad/s}$	1h
$\sigma_n = 100 \mu\text{s}$	$2.3 \cdot 10^{-2} \text{ rad/s}$	$7.6 \cdot 10^{-3} \text{ rad/s}$	2h 40min
$\sigma_n = 500 \mu\text{s}$	$4.6 \cdot 10^{-3} \text{ rad/s}$	$1.5 \cdot 10^{-3} \text{ rad/s}$	8h 50min

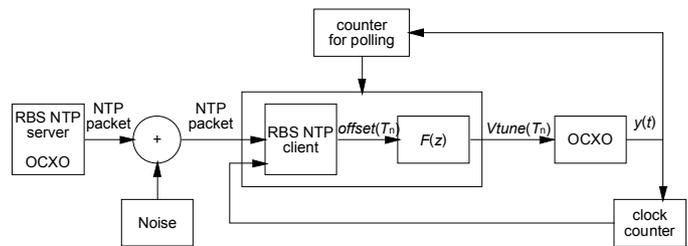


Fig. 3: Simulation model.

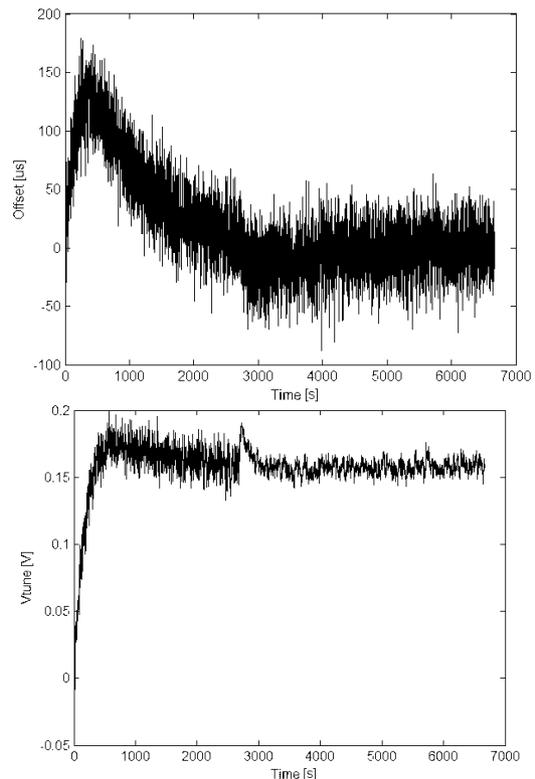


Fig. 4: $Offset(T_n)$ and $Vtune(T_n)$ with white Gaussian noise added to NTP reference time ($\sigma_n = 20 \mu\text{s}$, initial freq. offset $\Delta f = 10^{-6}$, sampling period $T = 1\text{s}$).

Let us notice that $Vtune(T_n)$ fluctuations around the mean value are proportional to the frequency error with respect to the input value. For example, in the second graph of Fig. 4, $Vtune(T_n)$ keeps approximately in range ± 0.005 V around the mean value after a transient of about 3000 s. This corresponds to ± 0.9 Hz of absolute frequency error (well below limits specified by ITU-R [12]).

Moreover, Table 1 presents some other performance figures, in terms of the time needed by the simulated system to converge within ± 1 Hz of frequency error, for various values of system parameters. Bandwidth values (-3dB) of fast and slow filters were chosen by optimizing system performance.

IV. EXPERIMENTAL PROTOTYPE

This section illustrates the experimental prototype realized by Prod-El for synchronizing two RBSs using Simple Network Time Protocol (SNTP) [20], a simplified version of NTP. The SNTP packet is identical to the NTP packet. Both NTP and SNTP use UDP/IP on door 123.

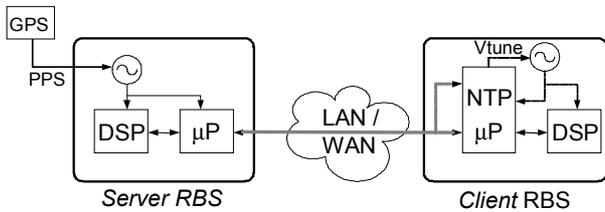


Fig. 5: Two RBSs connected via LAN or WAN.



Fig. 6: The prototype RBS module board implementing the NTP synchronization algorithm. Note the OCXO in the right low corner.

A. Outline of the Hardware and Firmware

The prototype is made of two identical board cards, one acting as SNTP server and one as SNTP client, as shown in Fig. 5. The board connects a microprocessor (μP) [21], a DSP, memory, an Oven Controlled Oscillator (OCXO), a LAN Ethernet controller and a LAN port. The board is connected via a 60-pins connector and a back panel to other boards (e.g.: transceiver, E1/T1 interface, 4-wire interface, power supply). A photo of the prototype board is shown in Fig. 6.

The SNTP client and server routines run on the microprocessor, over a Micro-Controller Linux (μCLinux) kernel. This is an open source Real-Time Operating System (RTOS), which works on μPs without Memory Management Units (mostly used in embedded systems). The RTOS manages the μP , a LAN Ethernet port, a Queue Serial Peripheral Interface (QSPI) connecting the μP to the DSP and a serial port.

The SNTP server uses GPS as time reference. A SNTP client does not distinguish between a NTP or SNTP server; unlike NTP clients, it tracks only one server.

The scheme of operation of the client board is shown in Fig. 7. The client exchanges NTP packets with the server by a polling interval $T = 1$ s (system sampling interval). When the client receives a packet, it calculates the Round Trip Time (RTT) and the system clock offset. These data may be passed to the function *adjtime()*, which adjusts the RTOS clock by small steps, or to the function *settimeofday()*, which resets it. The RTOS clock thus achieves a time accuracy of some μs .

This synchronization algorithm, nevertheless, proved not sufficient to attain satisfactory frequency stability. Thus, the system was improved. The output of functions, after digital-to-analog (D/A) conversion, tunes an OCXO. Therefore, the system works as a PLL, which is implemented partly by software

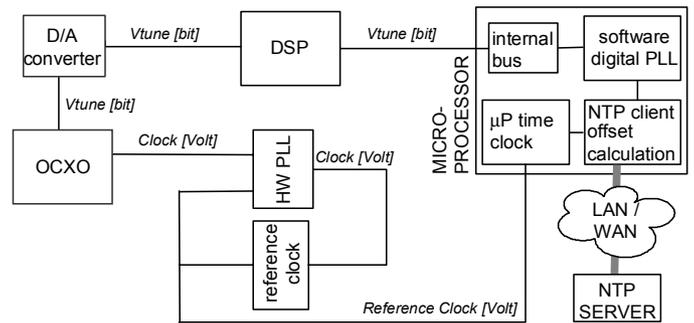


Fig. 7: Scheme of operation of the NTP client board.

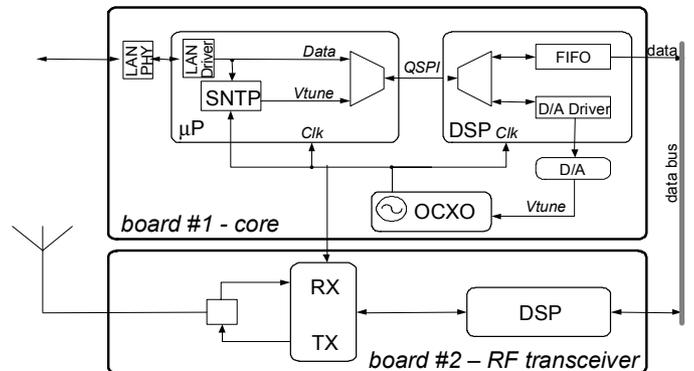


Fig. 8: Hardware block scheme of the RBS.

and partly by hardware. In summary, this system synchronizes in time the RTOS software clock and synchronizes in frequency the OCXO.

More in detail, the hardware block scheme of the RBS is outlined in Fig. 8, which shows two boards (board #1 - core and board #2 - RF transceiver) connected by a data bus.

In the board #1, the microprocessor first runs the LAN driver and the routine that distinguishes between data (e.g., voice, signalling, ...) and NTP packets. If an NTP packet is recognized, SNTP routines estimate V_{tune} , which is converted by the D/A to tune the OCXO. If a data packet is recognized, this is sent to the DSP, which converts the packet flow in a constant-bit-rate flow and vice versa. Data are then sent to other boards (e.g., in Fig. 8, the RF transceiver), not described because outside the scope of this paper.

The OCXO is the reference clock for the DSP, the microprocessor and the software clock of the NTP algorithm. It is also used to generate the RF carrier in the RF transceiver.

The RTOS software clock is controlled by the μP hardware frequency. Inside the μP , a timer is incremented at clock frequency. When the timer reaches a fixed value, an internal interrupt (*tick*) is generated by the RTOS, which increments the software clock. In this way, the software clock is directly controlled by the OCXO frequency.

SNTP evaluates the difference between the server and client clocks and then the time offset. After the SNTP client query, this information allows calculating the difference between the frequencies of the two software clocks and, proportionally, between the OCXO frequencies.

The flowchart of this algorithm is shown in Fig. 9.

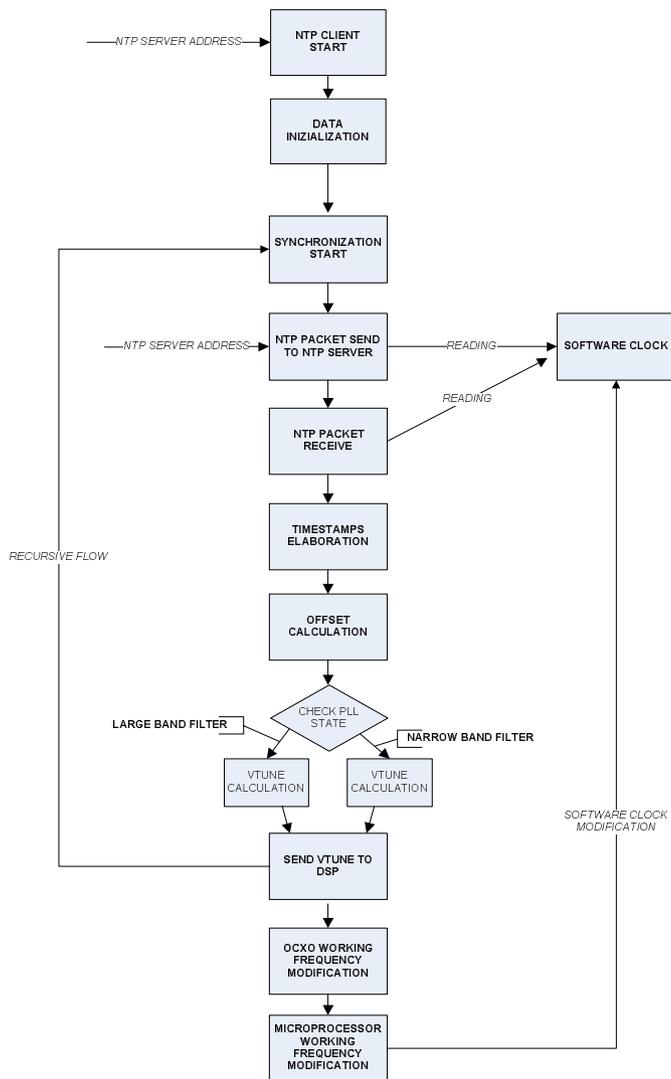


Fig. 9: Flowchart of the software synchronization algorithm.

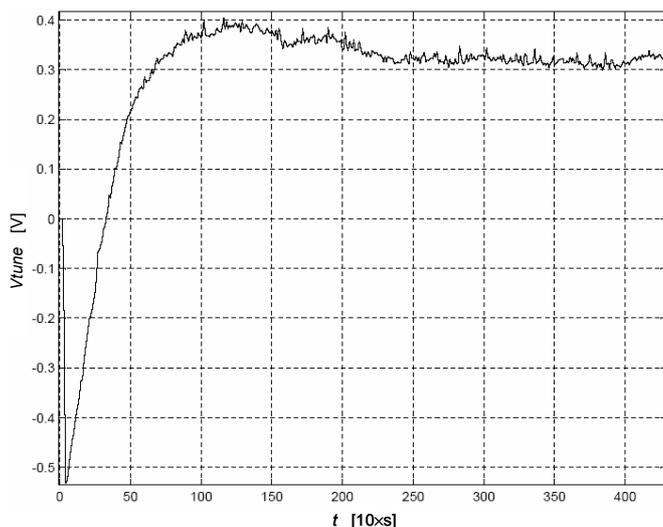


Fig. 10: Example of measurement results: V_{tune} measured after the D/A converter in the prototype system with a LAN between RBSs (Fig. 5).

B. Measurement Results

In this section, we present an example of preliminary measurement results obtained in our laboratory on the prototype boards described before.

The graph in Fig. 10 plots $V_{tune}(t)$ measured in the client RBS of a system configured as in Fig. 5, with a LAN between the two RBSs. In this graph, the level 0 V is the bias voltage between the D/A converter and the OCXO. The OCXO nominal frequency is 26 MHz. V_{tune} , after a transient of about 40 minutes, attains a mean value of about +0.32 V, equivalent to +56 Hz above the OCXO nominal frequency.

We note the excellent agreement with the simulation results shown in Fig. 4, where however higher noise was introduced to stress the system model. In measurement results shown in Fig. 10, on the other hand, fluctuations of $V_{tune}(t)$ around the mean value after the transient are very small, due to the controlled environment of the test bed (e.g., little packet jitter). In conclusion, the OCXO in the prototype client board converges to the reference frequency of the server board, as expected.

C. Implementation Pitfalls

A closer analysis and detailed measurements revealed an unexpected behaviour of the prototype system. Although the measured trend of $V_{tune}(t)$ in the client system (Fig. 10) shows that the slave OCXO converges to a stable frequency and that null time offset between RTOS clocks is achieved, as desired, the frequencies of the two OCXOs, when measured directly with a frequency meter, sometimes resulted actually slightly different.

The reason of this apparently contradicting behaviour was found by more in-depth measurements, by testing the system under different loads of the microprocessor, obtained by running different test programs besides NTP routines. As a matter of fact, the μ CLinux and microprocessor resulted not fast enough to handle all interrupts timely. In particular, the *tick* interrupt may be not always able to increment the RTOS clock at the time due, depending on simultaneous interrupts or different instructions run on the microprocessor.

As a result, the unwanted spurious delay between time offset calculation and OCXO frequency regulation may make the OCXO converge to a slightly different frequency, whilst time offset is kept null in phase-lock regime.

The obvious solution is to program V_{tune} computation routines on a faster DSP, rather than on the μ CLinux/microprocessor system. This is a mere implementation change, to solve a pitfall of the practical realization, which does not affect the validity of the overall system designed. This improvement work is currently in progress and will be featured in the next version of the system.

V. CONCLUSIONS

In this work, we studied a method to synchronize radio base stations in a simulcast network using NTP. In a simulcast network, both time and frequency synchronization is needed. In our system, the master RBS runs a SNTP server. All other RBSs run a SNTP client. According to the PLL scheme, we designed a system that synchronizes in frequency the OCXO

and in time the software clock of the client RBS, based on the time stamps read in SNTP packets received from the master RBS.

The system designed was modelled and simulated in Matlab Simulink[®], for feasibility and performance evaluation before designing a hardware prototype. All simulation results exhibited correct operation of the system designed and compliance to project requirements.

An experimental prototype was designed and realized in our laboratories. Measurement results on the prototype boards exhibited excellent agreement with simulation results and confirmed the practical feasibility of this method for application in commercial systems.

Nevertheless, a closer analysis and detailed measurements revealed some implementation pitfalls. As a matter of fact, the μ CLinux and microprocessor resulted not fast enough to handle all interrupts timely. In particular, the RTOS clock may be not always incremented at the time due, for example due to simultaneous interrupts. As a result, this unwanted delay may make the OCXO converge to a slightly different frequency.

Another potential pitfall of a practical realization of this synchronization scheme lies in the software clock resolution, which affects the time offset accuracy and hence the algorithm precision and convergence speed. This parameter depends on the RTOS implementation. The Unix OS have a microsecond clock resolution, which is acceptable given the synchronization requirements outlined in this paper. However, a faster convergence may be achieved with a nanosecond resolution.

The next version of the system described in this paper is currently under study. The main improvement will consist in programming PLL routines (time offset and *Vtune* computation) on a faster DSP, rather than on a microprocessor, to ensure that the RTOS clock is always adjusted at the time due, independently on the microprocessor load.

REFERENCES

- [1] S.Y. Mui, "A Simulcast UHF Terrestrial Radio Network", *IEEE Trans. Commun.*, vol. 42, no. 234, Feb./March/Apr. 1994, pp. 1460-1464.
- [2] S. Ariyavisitakul, T.E. Darcie, L.J. Greenstein, M.R. Phillips, N.K. Shankaranarayanan, "Performance of Simulcast Wireless Techniques for Personal Communication Systems", *IEEE J. Select. Areas Commun.*, vol. 14, no. 4, May 1996, pp. 632-642.
- [3] K. Dukhyun, G.L. Stuber, N. Hightower, "Performance of Simulcast Systems in Mobile Radio Environments", *Proc. IEEE 47th Veh. Technol. Conf.*, Phoenix, AZ, USA, May 1997.
- [4] J.M. Shea, K. Jung, "Simulcast Packet Transmission in Ad-Hoc Networks", *IEEE J. Select. Areas Commun.*, vol. 23, no. 3, March 2003, pp. 486-495.
- [5] R.J. Abelleiro, "Simulcast Voice and Data Communications system - Los Angeles County Sheriff California", *Proc. IEEE Conf. on Select. Topics in Wireless Commun.*, Vancouver, BC, Canada, June 1992.
- [6] ETSI, *Trans-European Trunk Radio, Documents 05.01, 05.02, 05.03, 05.04, 05.05, 05.08*, Nov. 1993.
- [7] ETSI TS 102 361-1/2/3 V1.2.1, *Digital Mobile Radio (DMR)*, Jan. 2006.
- [8] Bosch, BS 770 MPT 1327 VHF-UHF Repeater-Base Station, 1996.
- [9] S. Souissi, S. Sek, Hai Xie, "The Effect of Frequency Offsets on the Performance of FLEX(R) Simulcast Systems", *Proc. IEEE 49th Veh. Technol. Conf.*, Vancouver, BC, Canada, May 1999.
- [10] R. Petrovic, S. R. Filipovic, "Error Floors of Digital FM in Simulcast and Rayleigh Fading", *IEEE Trans. Veh. Technol.*, vol. 47, no. 3, Aug. 1998, pp. 954-960.
- [11] A. Morán, F. Pérez Fontán, J. M. H. Rábanos, M. Montero del Pino, "Quasi-Synchronous Digital Trunked TETRA Performance", *IEEE Trans. Veh. Technol.*, vol. 48, no. 2, May 1999, pp. 708-723.
- [12] ITU-R Rec. M.1077, *Multi-Transmitter Radio Systems Using Quasi-Synchronous (Simulcast) Transmission for Analogue Speech*, 1994.
- [13] S. Bregni, *Synchronization of Digital Telecommunications Networks*. Chichester, UK: John Wiley & Sons, 2002, 440 pp.
- [14] D.L. Mills, "Internet Time Synchronization: the Network Time Protocol", *IEEE Trans. Commun.*, vol. 39, no. 10, Oct. 1991, pp. 1482-1493.
- [15] D. Mills, RFC-1305 "Network Time Protocol (version 3). Specification, Implementation and Analysis", Network Working Group Report, University of Delaware, March 1992.
- [16] D. Mills, A. Thyagarajan, "Network Time Protocol Version 4 Proposed Changes", Electrical Engineering Department Report 94-10.2, University of Delaware, 1994.
- [17] D. Mills, A. Thyagarajan, B.C. Huffman, "Internet Time Keeping Around the Globe", *Proc. Precision Time and Time Interval (PTTI) Applications and Time Meetings*, Long Beach, CA, USA, Dec. 1997.
- [18] D. Mills, "Clock Discipline Algorithms for the Network Time Protocol v. 4", Electrical Engineering Report 97-3-3, University of Delaware, 1997.
- [19] D. Mills, "Time and Time Interval Measurement with Application to Computer and Network Performance Evaluation", Electrical Engineering Technical Memorandum, University of Delaware, Jan. 1996.
- [20] D. Mills, RFC-2030 "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI". Network Working Group Report, University of Delaware, 1996.
- [21] Motorola, "Mcf5282 Coldfire Microcontroller User's Manual - rev. 2".