Distributed Digital Locked Loops for Time/Frequency Locking in Packet-Based Wireless Communication

Umberto Spagnolini[†], Nicola Varanese[†‡], Osvaldo Simeone[‡], Yeheskel Bar-Ness[‡]
[†]Dipartimento di Elettronica e Informazione, Politecnico di Milano,
[‡]CWCSPR, New Jersey Institute of Technology
Email: {varanese, spagnoli}@elet.polimi.it,
osvaldo.simeone@njit.edu, barness@yegal.njit.edu

***Abstract*— In infrastructure-less wireless systems network-wise time and frequency synchronization can be achieved by exchanging mutual synchronization errors among neighboring nodes. Cooperative synchronization is based on the use of distributed digital locked loops (D-DLLs), as the extension to distributed systems of the classical concept of (analog or digital) looked loops. The convergence to a synchronized state depends ultimately on the degree of connectivity of the network. D-DLLs can be specialized for time or frequency synchronization by adopting an appropriate error detector, but preserving the same control loop. The focus of this paper is on distributed frequency synchronization for packet-based communication. A novel detector is proposed, which approximates the local mean frequency error from the uncoordinated transmission of packets by neighboring nodes. The performance of distributed frequency locked loops (D-FLLs) is evaluated for a wireless network employing packet-based cooperative relaying. Numerical validations are used to compare different frequency synchronization protocols in terms of speed of convergence and degradation of end-to-end performances.**

I. Introduction

Cooperative communications schemes are capable to enhance capacity and reliability of wireless networks. A typical underlying assumption is that all transmitted signals are synchronous, so as to enable simple signal modeling and receiver structures. However, synchronization errors cause performance degradation when employing virtual MIMO or distributed Space-Time Coding (STC) [1], thus calling for the design of effective synchronization techniques. Network synchronization strategies based on the iterative exchange of waveforms (e.g., wideband pulses [2]) over the wireless channel are receiving an increasing interest since they do not need any sort of centralized coordination. Timing synchronization at the physical-layer has been investigated in [3], by extending the concept and design practices of phase looked loops to distributed systems.

In this paper we propose a general framework of network-wise timing (symbol) and carrier frequency synchronization via distributed digital locked loops (D-DLL) as an extension of [3]. Rather than reconsidering the problem of symbol synchronization by pulse-coupled oscillators [3], here we focus on *carrier frequency synchronization* in packet-based cooperative communication system. A frequency tracking algorithm for systems with multiple transmitters and one receiver has been investigated in [4] but it entails a relevant increase in receiver complexity. Recently Parker et al. [5] narrowed the problem to a system with two-transmitters and one-receiver employing distributed STC (namely the Alamouti scheme). Their frequency synchronization protocol is basically a master-slave approach that does not easily scale to a larger number of nodes and relies on the assignment of specific pilot sequences to each node.

D-DLLs achieve a network-wise frequency-synchronous state for any (large) number of nodes without the need of an external master reference. Distributed frequency-locked loops (D-FLLs) employ the same control loop as generic D-DLLs, but necessitate a frequency error estimator (or detector in locked loops terminology) detector capable to measure from the superposition of transmitted packets the mean frequency error with respect to neighboring active nodes. The contribution of each active node to the detector output is proportional to the corresponding link quality, thus causing the convergence properties to depend on the overall network connectivity. To evaluate the performance of the algorithm, we consider a multi-stage relay network where multiple layers of relays forward information from a multi-antenna source to a single-antenna destination node, similarly to [6]. Each relaying stage encodes the information via a distributed Differential STBC (DSTBC) [7][8] (differential schemes have the advantage that their performance under frequency offsets is independent of the information block length). With the aid of simulation results, we compare different synchronization protocols applicable to this case study. In particular, it is shown that D-FLL is able to achieve the end-to-end Bit Error Rate (BER) of a fully synchronous system after a small number of iterations.

II. System model

Let us consider a network of N nodes employing packet-based communication (Fig. 1). We assume that the nodes are already frame-synchronous, and that within each frame a node can either listen or transmit a packet (half-duplex constraint), or just stay idle, depending on the specific communication and medium access protocols employed. Given a pre-established frame (or *large-grain*) timing synchronization,

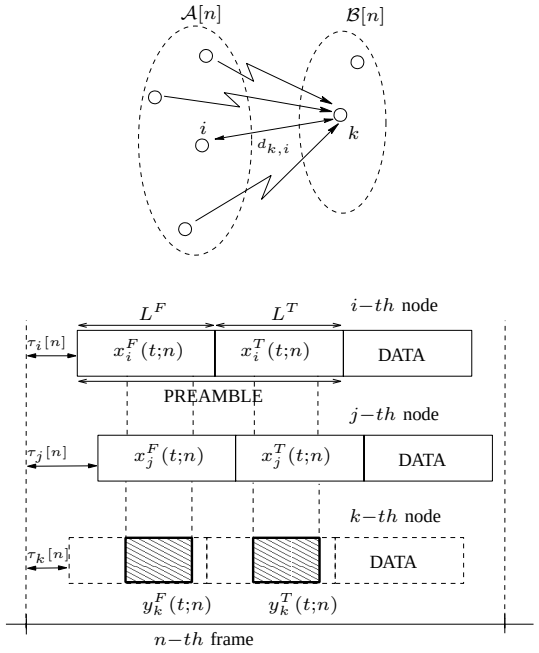

Fig. 1. Transmitted packets ($i, j \in \mathcal{A}[n]$) and received signal ($k \in \mathcal{B}[n]$) with respect to the ideal frame reference for a wireless network (top).

the transmission of the n-th packet from the i-th node still retains a residual delay $\tau_i[n]$ with respect to the ideal frame reference as in in Fig. 1. Each transmitted packet contains a preamble for training and a data payload. The preamble is further divided in two sections: the first part for frequency synchronization $x_i^F(t;n)$, and the second part for symbol (*fine*) timing synchronization $x_i^T(t;n)$. During the n-th frame, each node belonging to the set $\mathcal{A}[n]$ transmits a packet, and the superposition of (interfering) packets is processed by the nodes in the receiving set $\mathcal{B}[n]$ in order to extract from the training signals the correction signals for synchronization. Based on the frame structure, each receiving node (say the $k-$th) can segment the superposition of received preambles into the signals for frequency $y_k^F(t;n) = \sum_{i\in\mathcal{A}[n]} h_{k,i}[n]x_i^F(t;n)$ and symbol $y_k^T(t;n) = \sum_{i\in\mathcal{A}[n]} h_{k,i}[n]x_i^T(t;n)$ synchronization, $h_{k,i}[n]$ being the channel between the pair of nodes (i,k). D-DLLs for time and frequency locking differ only for the specific error detector employed, which has to be designed depending on the properties of the two received training signals $y_k^F(t;n)$ and $y_k^T(t;n)$.

The base-band model of the preamble signals $x_i^m(t;n)$ (with $m = F, T$ to denote frequency or time preambles, respectively) is a sequences of L^m band-limited pulses modulating a sinusoid at the current local carrier frequency $f_i[n]$

$$x_i^m(t;n) = \sum_{l=0}^{L^m-1} c_{i,l}^m g(t - \tau_i[n] - lT_s)e^{j(2\pi f_i[n]t+\phi_i[n])}, \quad (1)$$

where $c_{i,l}^m \in \{-1,+1\}$ is the training sequence for frequency ($m = F$) or symbol ($m = T$) synchronization and $\phi_i[n] \sim \mathcal{U}(0, 2\pi)$ is an arbitrary phase. The signal $g(t)$ is a root-raised cosine pulse with bandwidth $1/2T_s$. Frame synchronization guarantees that timings $\{\tau_k[n]\}_{k=1}^N$ have moderately small relative offsets, say $|\tau_k[n] - \tau_j[n]| \leq 3 \div 5T_s$ for $k \neq j$. The local frequency $f_k[n]$ and timing offset $\tau_k[n]$ are varying frame-by-frame as they are adjusted by each node via the local locking loops as discussed below.

The wireless channel $h_{k,i}[n]$ is modeled as frequency-flat Rayleigh fading with power that depends on the geometric distance $d_{k,i}$ according to the decaying law $E[|h_{k,i}[n]|^2] = d_{k,i}^{-\alpha}$. Channel is reciprocal $h_{k,i}[n] = h_{i,k}[n]$ (due to time division duplexing) and can change independently from frame to frame. To simplify the analysis, all nodes transmit with the same power and propagation delays are negligible compared to the timing resolution of $g(t)$, i.e., T_s.

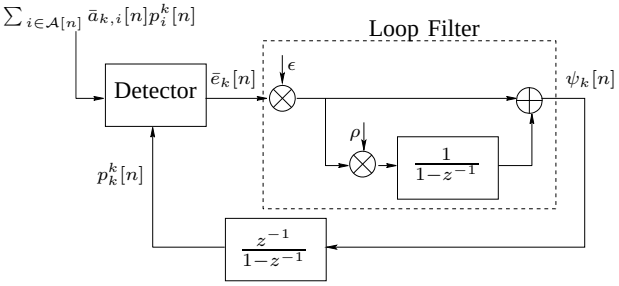

Fig. 2. D-DLL block diagram ($\bar{a}_{k,i} = a_{k,i}[n]/\sum_{i\in\mathcal{A}[n]} a_{k,i}[n]$) .

III. Distributed Digital Locked Loop (D-DLL)

Distributed frequency synchronization synchronization can be considered a special case of D-DLL [3]. The conditions for distributed network-wise synchronization can be decoupled into the design of D-DLL and the connectivity properties of the network. Both these aspects will be now revised in sequence.

Let $p_i^k[n]$ be the difference between the locking variable at the i-th node and the local reference at the k-th node (i.e., $p_i^k[n] = \tau_i[n] - \tau_{0,k}$ for symbol timing synchronization and $p_i^k[n] = f_i[n] - f_{0,k}$ for frequency entrainment, where $\tau_{0,k}$ and $f_{0,k}$ are the local timing and carrier frequency references), the loop control of the D-DLL for each node $k \in \mathcal{B}[n]$ is based on the combination of errors of $p_i^k[n]$ (for $i \in \mathcal{A}[n]$) with respect to local reference $p_k^k[n]$:

$$\bar{e}_k[n] = \frac{\sum_{i\in\mathcal{A}[n]} a_{k,i}[n]\left(p_i^k[n] - p_k^k[n]\right)}{\sum_{i\in\mathcal{A}[n]} a_{k,i}[n]} = \frac{e_k[n]}{\sum_{i\in\mathcal{A}[n]} a_{k,i}[n]}. \quad (2)$$

Even if weights $a_{k,i}[n]$ can be arbitrarily chosen (and are in practice varying on a frame-by-frame basis), a reasonable choice for the weights would be $a_{k,i}[n] = |h_{k,i}[n]|^2$, or $E[a_{k,i}[n]] = d_{k,i}^{-\alpha}$, thus giving more credit to offsets measured over more reliable channels (see also [3]). Based on the error $\bar{e}_k[n]$, the k-th node can update the local parameter according to the control loop of Fig. 2, namely

$$p_k^k[n] = p_k^k[n-1] + \psi_k[n-1] \quad (3a)$$

$$\psi_k[n] = \psi_k[n-1] + \epsilon(1+\rho)\bar{e}_k[n] - \epsilon\bar{e}_k[n-1], \quad (3b)$$

where ϵ and ρ are tunable fixed parameters. The update rule (3a)-(3b) describes a second-order discrete-time locked loop

where the locking variable is $p_k^k[n]$. Notice that for $\rho = 0$ the update rule (3a) is an instance of linear consensus algorithms, see, e.g., [9]-[10]. The choice $\rho \neq 0$ adds an integration path within the loop filter thus giving the possibility to track linear drifts in the local references or reduce convergence times. In D-DLL the error detectors approximate the metric (2) based on the noisy received training signals $y_k^F(t;n)$ and $y_k^T(t;n)$ for the frequency and time synchronization, respectively (see below).

For D-DLL with (2) the steady-state and dynamic analysis of the system with distributed control loops can be reduced to a linear second-order vector dynamic system. For fixed networks (i.e., $\mathcal{A}[n] = \mathcal{A}$, $\mathcal{B}[n] = \mathcal{B}$, and $a_{k,i}[n] = d_{k,i}^{-\alpha}$) with $\rho = 0$ the convergence properties depend only on network connectivity properties [3] [9] [10]. Faster convergence occurs when the network has a high degree of connectivity without presenting isolated (or almost isolated) clusters of nodes. In this respect, the performance driver is the inter-node distances $d_{k,i}$ that determines the geometric properties of the network (see [11]).

Recall that sets $\mathcal{A}[n]$ and $\mathcal{B}[n]$ are time-varying, since each node switches between transmission to reception modes according to some local rule. However, if the *switching sequence* $\{\mathcal{A}[n], \mathcal{B}[n]\}$ is periodic, the convergence analysis can be reduced to an *equivalent* network, with connectivity properties that depend on the network's geometry and the switching sequence adopted. In any case, convergence properties depend on the degree of connectivity within each switching period (the multi-hop protocol introduced in Sec. V shows that connectivity changes on a frame-by-frame basis).

A. A general framework for network synchronization

D-DLLs have a wide applicability to carry out network-wise synchronization procedures, being adaptable to a specific task simply by changing the detector design.

1) *Frame synchronization* is a coarse timing synchronization that ideally reduce the timing skewness down to the order of one (or few) symbol interval(s) T_s. To this end, one can use any of the methods to evaluate packet-wise timing errors (not covered here).

2) *Timing synchronization* is based on $y_k^T(t;n)$. Different users transmit the *same* training sequence $c_{i,l}^T = c_l^T$ so that synchronization state is achieved when all the sequenced transmitted by all the users are temporally aligned. It can be shown that, with a proper design of c_l^T (employing, e.g., a PN sequence), a conventional data-aided timing error detector can be employed within the D-DLL.

3) *Carrier frequency synchronization* is based on $y_k^F(t;n)$. Choosing the training sequence as $c_{i,l}^F = 1$, for $|\tau_k[n] - \tau_j[n]| \ll T_s$, or equivalently for L^F large enough, the transmitted signal for frequency training is equivalent to a single-tone

$$x_i^F(t;n) = e^{j(2\pi f_i[n]t + \phi_i[n])}, \tag{4}$$

where we assume $|f_i[n] - f_k[n]| \ll 1/T_s$. The received signal $y_k^F(t;n) = \sum_{i\in\mathcal{A}[n]} h_{k,i}[n] e^{j(2\pi f_i[n]t+\phi_i[n])}$ is a combination of sinusoids and error detector (2) measures the mean frequency error. Whereas several data-aided frequency difference detectors have been proposed in the past [12], it is not immediate to recognize their applicability to D-DLLs.

IV. Design of the frequency error detector

In this section we address the problem of designing an appropriate frequency difference detector that approximates (2) to realize D-FLLs under the assumptions on network and node operation as described above. The received RF signal is down-converted to baseband through the local frequency reference $f_{0,k}$, filtered by matched filter and sampled at frequency $1/T_s$. As a consequence, the sampled received signal at each node $k \in \mathcal{B}[n]$ is (to simplify, the frame index n is omitted from signals)

$$\tilde{y}_k^F(lT_s) = \sum_{i\in\mathcal{A}[n]} h_{k,i}[n] e^{j\left(2\pi f_i^k[n] lT_s + \tilde{\phi}_{k,i}[n]\right)} + \tilde{w}_k(lT_s), \tag{5}$$

where $f_i^k[n] = f_i[n] - f_{0,k}$, the phase $\tilde{\phi}_{k,i}[n] \sim \mathcal{U}(0, 2\pi)$ includes some timing errors, and noise $\tilde{w}_k(lT_s)$ is white Gaussian with power N_0. The received signals for frequency training after demodulation of (5) with the local offset $f_k^k[n] = f_k[n] - f_{0,k}$ are

$$y_k^F(lT_s) = \sum_{i\in\mathcal{A}[n]} |h_{k,i}[n]| e^{j\left(2\pi(f_i^k[n] - f_k^k[n])lT_s + \phi_{k,i}[n]\right)} + w_k(lT_s), \tag{6}$$

where $\phi_{k,i}[n] \sim \mathcal{U}(0, 2\pi)$ accounts for the overall phase offset, and $w_k(lT_s)$ is the Gaussian noise, still white with power N_0.

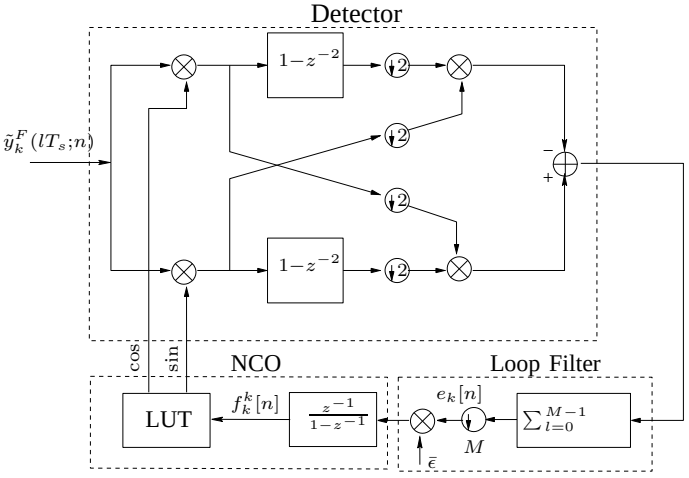

Fig. 3. Block diagram for a D-FLL employing a Digital Balanced Quadricorrelator (DBQC) as detector ($\rho = 0$). $\bar{\epsilon}$ is the loop gain ϵ normalized by the denominator in (8) (LUT, Look-Up-Table).

The frequency error detector has to extract the error signal (2) from $y_k^F(lT_s)$ in (6) (recall Fig. 2). If we let the training for frequency synchronization to have $L^F \to \infty$, the error (2) is the first moment of the energy spectrum of $y_k^F(lT_s)$. In particular, the frequency error $e_k[n]$ in (2) can be approximated

(for $L^F = 2M+1$ odd, with $M \geq 1$) as

$$e_k[n] \simeq \frac{1}{2\pi}\mathrm{Im}\Bigg\{\sum_{m=0}^{M-1} y_k^F((2m+2)T_s)y_k^{F*}((2m+1)T_s)+ \\ -y_k^F(2mT_s)y_k^{F*}((2m+1)T_s)\Bigg\}, \quad (7)$$

the normalized error (2) becomes

$$\bar{e}_k[n] \simeq \frac{e_k[n]}{2T_s\sum_{m=0}^{M-1}|y_k^F((2m+1)T_s)|^2}. \quad (8)$$

The expression in (7) can be shown to implement a Digital Balanced Quadricorrelator (DBQC) detector as depicted in Fig. 3. Analog BQC has been known for a long time to recover large frequency offsets (on the order of the symbol interval T_s) [12]. DBQC in Fig. 3 is the natural extension of the Analog BQC.

To summarize, Fig. 3 shows the D-DLL specialized for distributed frequency synchronization in infrastructure-less wireless network. It is worth to remark that for $|f_i[n] - f_k[n]| \ll 1/T_s$ and L^F sufficiently large, the detector (8) is equivalent to the linear detector in (2).

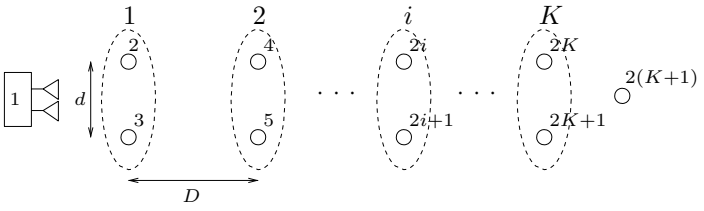

Fig. 4. K stages multi-hop multi-relay network.

V. Multi-hop multi-relay networks

The problem of frequency synchronization in a multi-hop multi-relay network (see Fig. 4) is helpful to illustrate the capabilities of D-FLLs. A two-antenna source node wishes to communicate to a single-antenna destination node. Since the destination is out of the transmission range for reliable reception from the source, K stages of two relay nodes aid the communication [6] with a total number of $N = 2(K+1)$ nodes. Hop-by-hop packet-based communication is performed where each transmitted packet contains a preamble signal to be employed for synchronization purposes. To isolate the impact of frequency synchronization, hereafter we assume perfect symbol and frame synchronization. To elaborate, during the first frame the source node transmits a packet (preamble and data) to the first stage of relays, which process it and forward it to the next stage during the second frame. In the i-th frame, the i-th relay stage process the signal received from the $(i-1)$-th stage, until the message finally gets to the destination, at the end of the $(K+1)$-th frame.

Targeting a scenario with frequency (and phase) offsets, here we consider the use of Differential STBC (DSTBC) at each relay stage (see Appendix A). DSTBC does not require channel estimation at the receiver side (at the price of about 3 dB loss as compared to coherent STBC) [8]. In addition, the probability of symbol error in the presence of carrier frequency offsets can be shown to be independent of the block (packet) length. According to the communication protocol, the $(i-1)$-th relay stage employs a DSTBC to forward the message to the i-th stage, where each relay node independently decodes (Decode and Forward relaying, DF) and re-encodes for transmission in the following slot. Different synchronization strategies can be devised for this scenario.

a - Open-loop technique. In this quite conventional strategy, the local offset $f_k^k[n]$ at the nodes of the receiving stage is computed upon reception of the preamble signal from the transmitting stage employing (8). Namely, each node adjusts its frequency in a *memoryless* (one-shot, or open-loop) fashion according to the instantaneous measurement only when it decodes the data payload of the transmitted packet. This scheme essentially assumes that the previous stages have already achieved a good level of synchronization. However, for small L, the one-shot frequency estimate is affected by a relevant error already at stage 1 that propagates to the following stages in the subsequent steps.

b - Closed-loop technique A. Similarly to the open-loop technique above, only the nodes in the receiving stage update their local offsets. However, a running frequency estimate is performed according to (3a)-(3b), (8) (or equivalently Fig. 2-3). Again, here each node updates its offset only when it is effectively receiving data, that is the set $\mathcal{B}[n]$ of Sec. II contains *only* the receiving stage of relays (or the final destination), i.e., $|\mathcal{B}[n]| = 2$ or 1. The local frequency is updated by combining the new estimate provided by the detector (8) with previous estimates, thus allowing for a progressive refinement of the local parameter value.

c - Closed-loop technique B. In a large collaborative strategy *all* the nodes that are currently not transmitting (except the source) update their running frequency according to Fig. 2-3. This scheme extends the previous technique by taking into account that the distributed algorithm (3a)-(3b) shows faster convergence times in well-connected networks. The simplest way to improve the network connectivity in each frame (for synchronization purposes) is to let all the nodes listen to each synchronization signal transmitted in the network, whether or not they are the final destination of the data payload.

VI. Numerical results

In this section we compare the impact of the synchronization schemes introduced in Sec. V on the performance of the network in Fig. 4.

Each transmitted symbol has unitary power and each link is affected by additive white Gaussian noise with variance N_0. The network SNR is defined as $SNR = 1/N_0$. The path loss is $\alpha = 3$. The preamble signal is always assumed to be transmitted with a 5 dB power boost with respect to the information-bearing part of the packet. The employed modulation is BPSK. The local carrier frequency reference at the k-th node $f_{0,k}$ is random and it is assumed to be uniformly distributed in the interval $[-f_{0,\max}, f_{0,\max}]/T_s + f_0$, f_0 being

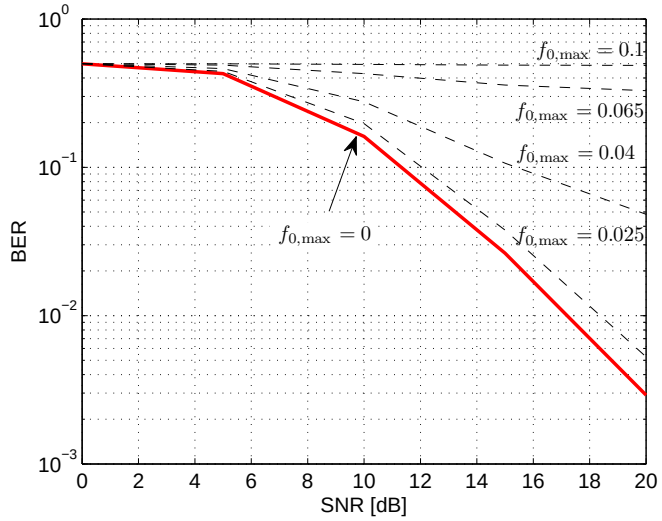

Fig. 5. End-to-end BER after $K = 5$ stages for the network in Fig. 4 without frequency offset compensation ($\epsilon = 0$, $D/d = 1.2$).

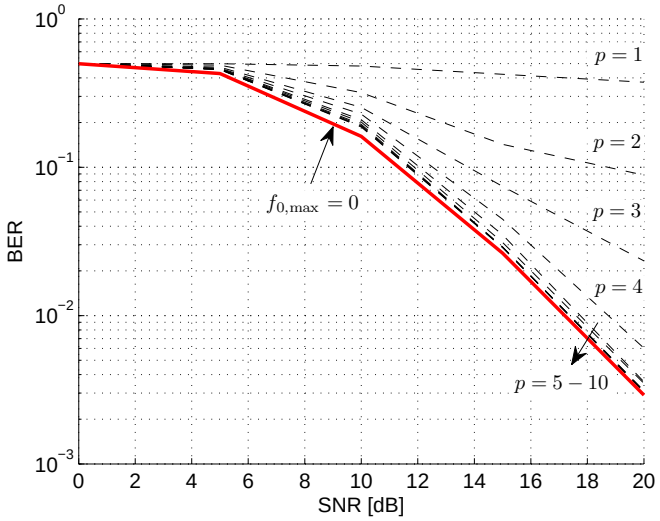

Fig. 6. End-to-end BER improvement for the closed-loop algorithm B ($f_{0,\max} = 0.15$, $K = 5$, $D/d = 1.2$, $\epsilon = 0.35$, $L^F = 11$).

the nominal carrier frequency of the communication system. As a measure of network-wise frequency synchronization, we use the mean deviation $\xi[n]$ of the frequencies $f_k[n]$, where $\xi^2[n] = 1/N \sum_{k=1}^{N} E[(f_k[n] - 1/N \sum_{k=1}^{N} f_k[n])^2]$, and the expectation is taken over different realizations of the initial frequencies, channels and noise. The ratio between the inter-stage distance and the intra-stage distance is $D/d = 1.2$. As we assume that local carrier references are not drifting in time, we set $\rho = 0$ for closed-loop techniques (first order D-DLL). Also, trading off lock accuracy versus convergence speed, we set the loop gain $\epsilon = 0.35$. Finally, the training length is $L^F = 11$ samples for both open and closed-loop techniques.

Fig. 5 shows the degradation in the end-to-end BER due to increasing frequency offsets among the nodes in the network, in the case where no frequency offset correction takes place ($\epsilon = 0$). In this case $K = 5$ stages. In ideal conditions ($f_{0,\max} = 0$), the communication scheme provides a diversity gain of 2, while a maximum spread $f_{0,\max} = 0.04$ (corresponding to $\xi[0] = 2.2 \cdot 10^{-2}$) is sufficient to nullify the diversity gain of the multi-relay scheme, raising the slope of the curve from 2 to approximately 1 (for this range of SNR values).

In the following simulation results, we consider the end-to-end transmission of p packets, corresponding to $n = (K + 1)p$ frames, and evaluate the corresponding mean frequency deviation $\xi[p]$ and the end-to-end BER associated with each packet. In Fig. 6, the BER obtained with the closed-loop scheme B is shown varying the number of transmitted packets p ($K = 5$ stages, maximum spread $f_{0,\max} = 0.15$). It is seen that $p = 4$ packets are sufficient to yield a sufficient degree of synchronization that entails a negligible loss as compared to perfect synchronization (at least for this range of SNR values).

Fig. 7 compares the speed of convergence of the three algorithms discussed in the previous section, in terms of the mean frequency deviation $\xi[p]$, ($K = 3$ stages, $f_{0,\max} = 0.15$ and $SNR = 15dB$). The open loop approach is limited by

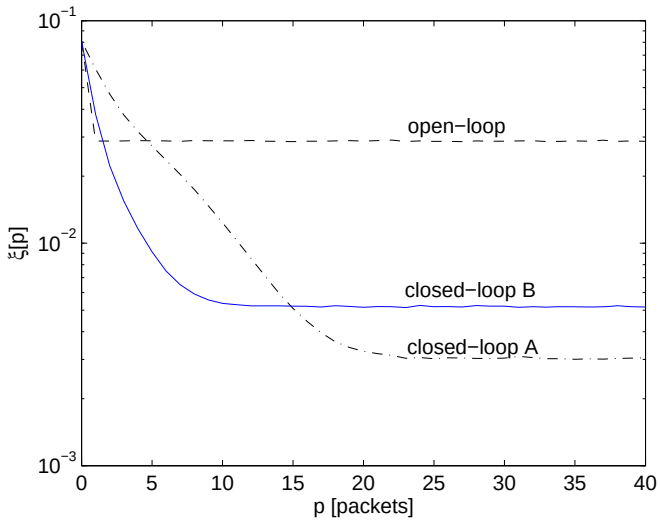

Fig. 7. Mean deviation of the frequencies $f_k[n]$ for different algorithms ($f_{0,\max} = 0.15$, $K = 3$, $D/d = 1.2$, $\epsilon = 0.35$, $L^F = 11$, $SNR = 15dB$).

the very few samples employed ($L^F = 11$), whereas both the closed-loop algorithms have an error floor which is due to additive noise at the output of the detector. The algorithm B converges faster, but at the price of a higher noise floor. However, this impairment is immaterial to BER performance (see below). Algorithm B achieves faster convergence times essentially because the equivalent graph has better connectivity properties. Also, scheme B is limited by a higher noise floor as it causes more noise to be exchanged among the nodes.

Finally, in Fig. 8 we verify the impact of the performance

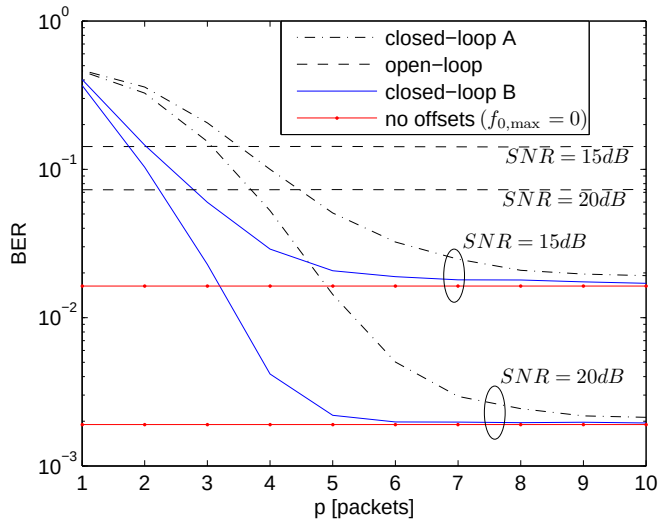

Fig. 8. End-to-end BER for different algorithms ($f_{0,\max}=0.15$, $K=3$, $D/d=1.2$, $\epsilon=0.35$, $L^F=11$, $SNR=15,20dB$).

in Fig. 7 on the end-to-end BER as a function of the packet index p ($K=3$ stages and $f_{0,\max}=0.15$). Despite of a higher error floor, the algorithm B needs only $p=4$ packets to get close to the synchronous system performance for both SNR values, while the scheme A requires at least $p=7$ packets. The open loop technique would need a longer preamble sequence to improve the performance.

VII. Conclusions

In packet-based wireless communication, distributed digital locked loops (D-DLLs) are an effective solution to attain network-wise synchronization, without any master reference. In this paper we introduced an application of D-DLLs, whereby a state of (symbol) timing and frequency synchronization can be achieved by the uncoordinated exchange of training signals among nodes. In particular, we proposed a novel frequency detector for distributed frequency synchronization employing D-DLLs. Further, a viable integration of the proposed algorithm has been studied within a multi-stage multi-relay network employing DSTBC and packet-based communication. The distributed synchronization procedure has been shown to be able to mitigate the effect of frequency offsets more efficiently than other open and closed loop techniques.

Appendix A: DSTBC with frequency offsets

Without loss of generality, here we consider the the l-th Space-Time codeword transmitted during the n-th frame from the i-th stage to the $(i+1)$-th stage (nodes $2i$ and $2i+1$ are transmitting, see Fig. 4). Also, we focus on the processing at node $2i+2$ within the $(i+1)$-th stage, as the two receiving nodes decode independently of each other.

Let the input alphabet $\mathscr{X}=\{\mathbf{X}_l\}$ be a finite set of 2×2 unitary matrices. The differentially encoded Space-Time codeword actually transmitted over the channel is $\mathbf{W}_l=\mathbf{W}_{l-1}\mathbf{X}_l$, with $\mathbf{W}_0=\mathbf{I}$. As suggested in [8], $\mathbf{X}_l$ is chosen as a *normalized* Alamouti code matrix, such that $\mathbf{X}_l^H\mathbf{X}_l=\mathbf{I}$. Assuming a synchronous system, the 1×2 received vector signal over two consecutive symbol periods is

$$\mathbf{y}_l = \mathbf{h}\mathbf{W}_l + \mathbf{n}_l = \mathbf{y}_{l-1}\mathbf{X}_l - \mathbf{n}_{l-1}\mathbf{X}_l + \mathbf{n}_l, \quad (9)$$

where $\mathbf{h}=[h_{2i+2,2i}[n], h_{2i+2,2i+1}[n]]$ is the channel between the transmitting nodes $(2i, 2i+1)$ and the receiving node $2i+2$ (constant over the whole frame period), and the additive noise $\mathbf{n}_l\sim\mathcal{CN}(\mathbf{0},N_0\mathbf{I})$. From (9), as in differential modulation for point-to-point channels, the signal vector received at time $l-1$ is the effective channel at time l, and the information-bearing signal is corrupted by two noise terms.

In case of different frequency offsets at the two transmitting nodes, the received vector signal (9) can be written as

$$\mathbf{y}_l=\mathbf{h}\begin{bmatrix} w_{1,l}e^{j\omega_1 2lT_s} & -w_{2,l}^* e^{j\omega_1(2l+1)T_s} \\ w_{2,l}e^{j\omega_2 2lT_s} & w_{1,l}^* e^{j\omega_2(2l+1)T_s}\end{bmatrix}+\mathbf{n}_l, \quad (10)$$

where $\omega_1=2\pi(f_{2i}[n]-f_{2i+2}[n])$ and $\omega_2=2\pi(f_{2i+1}[n]-f_{2i+2}[n])$ are the offsets between the two transmitting nodes and the receiving node. Due to the different carrier frequencies, the effective Space-Time codeword is no more orthogonal, generating Inter-Symbol-Interference at the output of the detector.

References

[1] J. Mietzner, J. Eick, and P. A. Hoener, "On distributed Space-Time Coding techniques for cooperative wireless networks and their sensitivity to frequency offsets," in *Proc. of 2004 ITG Workshop on Smart Antennas*.

[2] Y. W. Hong and A. Scaglione, "A scalable synchronization protocol for large scale sensor networks and its applications," *IEEE J. Select. Areas Commun.*, vol. 23, p. 10851099, May 2005.

[3] O. Simeone and U. Spagnolini, "Distributed synchronization for wireless sensor networks with couple discrete-time oscillators," in *Eurasip Journal on Wireless Commun. and Networking*, vol. 2007, July 2007, pp. 3153–3167.

[4] D. Veronesi and D. L. Goekel, "Multiple frequency offset compensation in cooperative wireless systems," in *Proc. of IEEE GLOBECOM 2006*.

[5] P. Parker, P. Mitran, D. W. Bliss, and V. Tarokh, "On bounds and algorithms for frequency synchronization for collaborative communication systems." [Online]. Available: http://arxiv.org/PS_cache/arxiv/pdf/0704/0704.3054v1.pdf

[6] S. Borade, L. Zheng, and R. Gallager, "Amplify-and-forward in wireless relay networks: rate, diversity, and network size," *IEEE Trans. Inform. Theory*, vol. 53, pp. 3302–3318, Oct. 2007.

[7] S. Yiu, R. Schober, and L. Lampe, "Distributed space-time block coding," *IEEE Trans. Commun.*, vol. 54, pp. 1195–1206, July 2006.

[8] E. G. Larsson and P. Stoica, *Space-time block coding for wireless communications*. Cambridge Univeristy Press, 2003.

[9] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Automat. Contr.*, vol. 49, pp. 1520–1533, Sept. 2004.

[10] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, pp. 215–233, Jan. 2007.

[11] O. Simeone, U. Spagnolini, G. Scutari, and Y. Bar-Ness, "Physical-layer distributed synchronization in wireless networks and applications," *Physical Communication*, pp. 67–83, Mar. 2008.

[12] U. Mengali and A. N. D'Andrea, *Synchronization Techniques for Digital Receivers*. Springer, 1997.