Bio-inspired Proximity Discovery and Synchronization for D2D Communications

Shih-Lung Chao, Hsin-Ying Lee, Ching-Chun Chou, and Hung-Yu Wei *Member, IEEE*

***Abstract*— In this paper, a distributed mechanism for application-aware proximity services (ProSe) in device to device (D2D) communications is presented. The method, which is derived from the bio-inspired firefly algorithm, can achieve proximity discovery and synchronization at one time. To be precise, this mechanism not only enables neighbour discovery and service discovery simultaneously, but achieves synchronization in physical communication timing and service interests in the meanwhile. However, the basic firefly algorithm is not perfectly suitable for larger scale systems such as LTE-A D2D. In most network topologies, the property that each node may not be able to hear all its neighbours makes the basic version fail. Thus, we further propose the firefly spanning tree (FST) algorithm which can solve the topology-dependent divergence problems.**

***Index Terms*— D2D, bio-inspired algorithm, application-aware services, proximity discovery, synchronization.**

I. Introduction

Device-to-device (D2D) communication is an emerging technology. This technology enables a wireless communication device to connect to another device directly. Recently, 3GPP started discussing proximity based services (ProSe)[1][2] for D2D communication in LTE. Proximity discovery is a key component for D2D communication. Take public safety application as an example, devices are usually distributed in a broad area as shown in Fig. 1. Infrastructure nodes may be unavailable due to the damage from a disaster. When some devices are outside the base station's (BS) coverage, a distributed mechanism is required for devices to achieve proximity discovery and synchronization. The proximity discovery in general can be categorized into two different problems, physical communication[8] and application level discovery. The integration of two proximities is desired because the signaling procedures can be reduced. For example, physical-level proximity discovery requires signal exchange among devices. Application-level proximity discovery requires the devices to find other devices with the same interests. Thus, integration of proximity discovery in view of physical communication and application is needed.

Previous works applied this sync method in different manners. Wener-Allen *et al.* implemented decentralized RFA (reachback firefly algorithm) on TinyOS-based motes, and provided theoretic improvement[4]. Tyrrell *et al.* studied an exquisite design[5] and its application[6]. Lucarelli and Wang

Manuscript received XXX XX, 2013; The associate editor coordinating the review of this letter and approving it for application was XXX

The authors are with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan (Corresponding author e-mail: hywei@cc.ee.edu.ntu.tw).

Digital Object Identifier XXXXXXXXXX

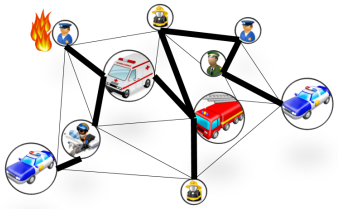


Fig. 1. An example of a firefly spanning tree (each edge denotes a possible connection between two edges) of a D2D graph. By selecting the (heavy) edges by the proposed algorithm, the tree spanning every device makes the network synchronized.

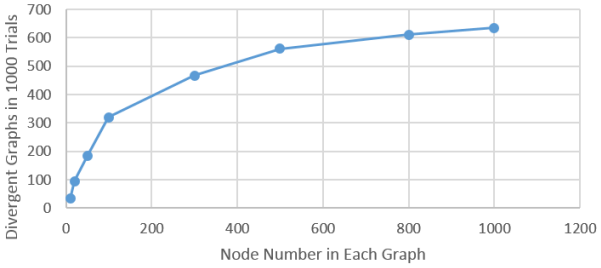


Fig. 2. The simulation results of applying basic firefly algorithm on graphs with different scales. The test cases are randomly generated graphs with 10, 20, 50, 100, 300, 500, 800 and 1000 nodes. Each dot indicates the divergence times within 1000 trials for each test case. The detailed parameters are given in Table I.

described the general model, and the convergent condition[7]. Nevertheless, in all conventional work above, the topology (physical layer radio connectivity topology) model is always idealized.

Our LTE-A D2D simulator is constructed based on the LTE simulator [9]. It includes a link level simulator and a system level simulator. As observed in simulations, the network topology significantly affects the convergence. As shown in Fig. 2, if we naively apply the basic firefly algorithm in randomly generated topologies, a substantial ratio of systems are even unable to achieve the state sync. In current systems, like D2D communications in LTE-A, the significant growth of involved devices makes the complexity of the network marking increase. This implies that the basic firefly algorithm is no longer suitable to emerging communications. Thus, it is imperative to create a new replacement which is adaptable to all kinds of topology. In this paper, a distributed topology-adaptive algorithm named firefly spanning tree (FST) is proposed. FST is an algorithm based on the basic firefly algorithm and a spanning tree algorithm, it is able to transform an divergent graph into a convergent graph as shown in Fig. 1. In simulation, the algorithm not only preserves all the

benefits from the original one, additionally it outperforms other common sync methods [13] under the LTE-A D2D circumstance.

II. Basic Firefly Algorithm

In the real world, synchronization between one another is one of the the largest problems that a swarm of fireflies have to face every day. This is a totally self-organized synchronization problem, resembling the one we face in communications. Surprisingly, by simply applying their innate flashes, the tough problem has an elegant solution.

Basically, the synchronous flashing progress can be modelled as the behaviour of a population of identical integrate-and-fire oscillators. It can be expressed as a phase function $\phi_i(t)$ which is integrated from zero to a certain threshold ϕ_{th}. When ϕ_{th} is reached, the oscillator flashes, or "fires." After firing, $\phi_i(t)$ is reset to zero. If not coupled with others, i.e. no firing signal is detected, the oscillator will naturally fire with a period equal to T. This may be described as $\frac{d\phi_i(t)}{dt} = \frac{\phi_{th}}{T}$.

When coupled to others, an oscillator is receptive to the firing signal of its neighbours. When a given oscillator fires, it pulls the others up by a fixed amount ϵ, or brings them to ϕ_{th}, whichever is less, i.e., $\phi_i(t) = \phi_{th} \Rightarrow \forall j \neq i : \phi_j(t^+) = min(\phi_{th}, \phi_j(t) + \epsilon)$. If the threshold ϕ_{th} is normalized to 1, when a node j fires at $t = \tau_j$, the above equation can be transformed into a piecewise linear function, $\phi_i(\tau_j) + \Delta\phi(\phi_i(\tau_j)) = min(\alpha\phi_i(\tau_j) + \beta, 1)$ with $\alpha = e^{b\epsilon}$ and $\beta = \frac{e^{b\epsilon}-1}{e^b-1}$, where $\Delta\phi$ and b denote the phase increment function and the dissipation factor, respectively.

It has already been proven that, provided that certain constraints (e.g. $\alpha > 1$, $\beta > 0$, and full meshed) on the coupling between entities are met, for an arbitrary number of entities and independent of the initial condition, the network always synchronizes. Plenty of aspects such as the effect of propagation delay, channel attenuation, and noise have also been addressed in the literature [5] [6]. Optimal selection of the parameters has also been completely discussed in some solid works. Thus, the scope of this paper will not cover the above issues again.

III. Proximity Discovery and Synchronization

The basic concept of firefly algorithm can be applied to D2D networks as shown in Fig. 3. In this paper, we proposed a distributed mechanism to achieve proximity discovery and synchronization. This mechanism enables neighbour discovery and service discovery simultaneously. In addition, it also achieves synchronization in physical communication timing and service interests in our application.

The mechanism achieves proximity discovery by sending and detecting proximity signals (PSs) among devices as shown in Fig. 4. Each device possesses a counter and broadcasts PSs periodically. The counter value increases by time and a threshold is set for the counter. Devices detecting the PSs will increase the counter value by a fixed rate. Once the counter reaches the threshold, the device will broadcast the PS, and then reset the counter back to the initial value like zero. The PS is broadcast to the neighbouring devices, and

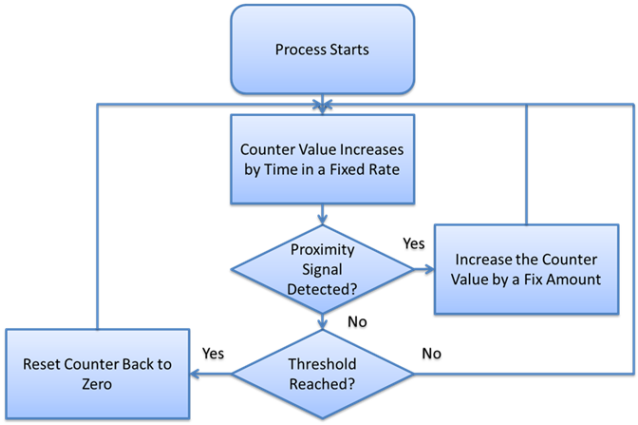
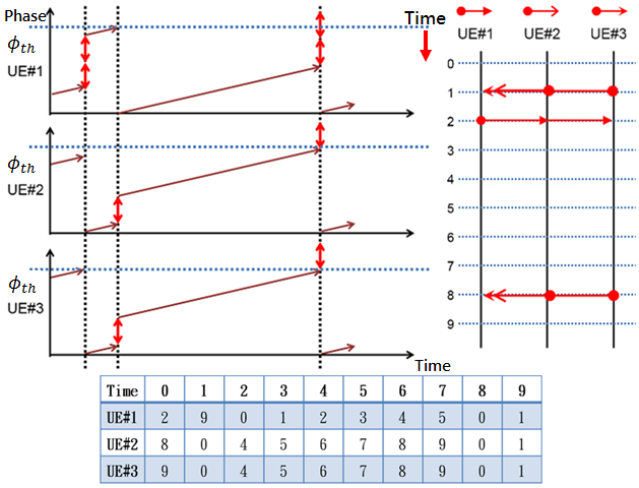


Fig. 3. The flowchart of firefly algorithm.



Time	0	1	2	3	4	5	6	7	8	9
UE#1	2	9	0	1	2	3	4	5	0	1
UE#2	8	0	4	5	6	7	8	9	0	1
UE#3	9	0	4	5	6	7	8	9	0	1

Fig. 4. The signaling flow of D2D UEs' synchronization. The timing of UE#1∼UE#3 sending proximity signals (PSs) were presented in the figure. As the UE's counter exceeded the threshold, the UE will broadcast the PS, and the UEs receiving the signal will increment its counter. After several repetitions, the synchronization is finally reached once and for all.

those devices increase their counter values accordingly. This process of PS broadcast and detection will continue until all the devices achieve synchronization in the period of signal broadcast. Note that the rate of increment and threshold may be different for different applications/services/groups. Take LTE-A system as an example, predetermined RACH codes (or other CDMA codes) or preambles can be used to transmit the PS in the proposed scheme. Different codes indicate different application interests. This implies that a device may transmit multiple codes to indicate its interest in these multiple applications. Transmission in different radio transmission opportunity (e.g. RACH resource) can also be used to indicate different application interest.

After synchronization is reached within the group, several communication activities can be done. For example, the mechanism might be served as a keep-alive signaling method. It can also be directly applied to activate data session between devices. For instance, the PS may use two different codes (e.g. a pair RACH code). One code indicates standby and continue to the sever as the synchronization/keep-alive purpose. The

other code triggers other events. We further assume that user equipments are full-duplex devices which are capable of simultaneously transmitting and receiving proximity signals. LTE-A RACH preambles are OFDM symbols, and different preambles may be transmitted in parallel without inter-group interference. However, intra-group proximity signal interference may still arise due to collision or misalignment. Such interference will influence the counter value increment. That is, devices may only detect 1 preamble while more than two devices are sending the proximity signal. Firefly algorithm still holds in this case.

For the purpose of reducing the signaling process in both physical and application levels of proximity discovery, we introduce the concept of firefly synchronization into D2D communications. The properties of the algorithm are quite suitable to an idealized system, like a full meshed network. However, in real systems, an idealized topology is not a common case. To our best knowledge, currently there is no such research that focuses on applying similar algorithms on any randomly generated topology.

IV. Firefly Spanning Tree

A D2D network can be formulated into a graph $G(V, E)$, where vertices V are independent devices and edges E are communication links. Links can be weighted by the strength of the PS. For preserving all the benefits of the mechanism in any randomly generated topologies, first we need to find a basic structure which is able to sustain the state of synchronization. By conducting several experiments, we found that the structure of trees may be a good candidate. In theorem 1, the stability of trees can be verified by mathematical induction.

Theorem 1: For any acyclic graph (i.e., tree, connected graph without simple cycles), by applying the firefly algorithm, the synchronization of nodes is always achieved and sustained.

Proof: Mathematical induction can be used to prove that the above statement in the theorem always holds.

The basis: Show that the statement holds for the node number $n = 2$. Based on [3], when there are only two nodes (i.e. full meshed network with two nodes), the system can be synchronized. **The inductive step**: Show that if the trees with k nodes (i.e. $n = k$) hold, then any acyclic structures with $n = k + 1$ nodes will also hold.

A new tree T' with $k + 1$ nodes can be achieved by inserting a new node L to a k-node tree T. One new connection is established between L and any of the nodes P in T. Since $\phi_P(t) = \phi_L(t), \forall P, \exists t$, the derivation can be divided into three conditions based on the impact of phase shift $\phi_P(t)$ from $\phi_{T-\{P\}}(t)$ after the insertion of L:

1) **No phase shift occurs (i.e. $\phi_P(t) = \phi_{T-\{P\}}(t)$)**: this means that the synchronizing situation of T' is equal to T. That is, the convergence is immediately completed.
2) **Phase shift occurs (i.e. $\phi_P(t) \neq \phi_{T-\{P\}}(t)$), and P reaches the threshold**: P receives a PS from $T - \{P\}$ at t_0, which means the entering of L makes P temporarily unsynchronized from $T - \{P\}$. If the PS makes P saturate (i.e. $\alpha\phi_P(t_0) + \beta \geq 1$) , L will also be lifted to saturation by P's PS since $\phi_P(t_0^-) = \phi_L(t_0^-)$. As a result, the condition of T' is equal to T.
3) **Phase shift occurs, but P does not reach the threshold**: If the PS at t_0 does not make P saturate (i.e. $\phi_P(t_0^+) = \alpha\phi_P(t_0) + \beta < 1$), P and L are temporarily unsynchronized. At the moment just before P's next saturation, say t_1, the phase of L is $\phi_L(t_1^-) = \phi_L(t_0) + (1 - (\alpha\phi_P(t_0) + \beta)) = (1 - \alpha)\phi_P(t_0) + (1 - \beta)$. After that, the PS from P is received by L, we have: $\phi_L(t_1^+) = min(1, \alpha((1 - \alpha)\phi_P(t_0) + (1 - \beta)) + \beta)$. Since $\alpha > 1$ and $\alpha\phi_P(t_0) + \beta < 1$, we have $\alpha((1 - \alpha)\phi_P(t_0) + (1 - \beta)) + \beta - 1 = (1 - \alpha)(\alpha\phi_P(t_0) + \beta - 1) > 0$, which means $\phi_L(t_1^+) = min(1, \alpha((1 - \alpha)\phi_P(t_0) + (1 - \beta)) + \beta) = 1 = \phi_P(t_1^+)$. As a result, the condition of T' is equal to T.

Since both the basis and the inductive step have been performed, by mathematical induction, the statement always holds. ■

After finding a suitable structure, we still need a practical communication algorithm for transforming a network with random topology into an acyclic tree. A distributed algorithm for solving this problem is proposed. Inspired by the well-known GHS algorithm[11], we propose the **Firefly Spanning Tree (FST) Algorithm** which involves the construction of a robust spanning tree with the strongest signal strength on graphs in a fully distributed manner. It is radically different from the classical sequential problem, yet the most basic still approach resembles the well-known Boruvka's algorithm.

At the beginning of the algorithm, devices know only the weights of the links which are connected to them. After the time of linearithmic scale (i.e. the asymptotic time complexity is $O(VlogV)$), as the output of the algorithm, every device knows which of its links belong to the FST and synchronizes with the remaining neighbours. Note that the process involves two different proximity signals. PS_H is used for the synchronization between subgraphs, and PS_G is used for the regular operation of the basic firefly algorithm throughout the network. The pseudocode of FST algorithm is as described in Algorithm 1 and 2. Basically, FST is designed based on the two properties: greedy choice property and optimal substructure. By applying the two-level firefly algorithm and a greedy algorithm, we can combine all the available subgraphs into one spanning tree, i.e. the robustness of FST can be guaranteed. That is, the sum of signal strength, i.e. the weight of FST, is greater than or equal to the weight of every other spanning tree. The exact validity of FST can be mathematically proved by contradiction.

Algorithm 1: Firefly-Spanning-Tree(G)

```
for each v ∈ G.V do
    S_v ← Make-Subgraph(v);
FST ← {S_v| v ∈ G.V};
/* each S_v ∈ S performs basic firefly algorithm (using PS_G) and the following
   procedures in parallel */
while |FST| ≠ 1 do
    if Head-Connect(S_v.Head, S_v, FST) = true then
        Sync-in-Subgraph(S_v, S_{u≠v});
        /* Apply basic firefly algorithm (using PS_H) to sync the heads of S_u
           and S_v.*/
    else
        Change-Head(S_v);
        /* other than the current head, choose v ∈ S_v with largest edge ∉ S_v
           as the new head of S_v */
        continue;
    Merge-Subgraph(S_v, S_u, FST);
    /* S_v ← S_v ∪ S_u, delete S_u from S, and choose v ∈ S_v with largest
       edge ∉ S_v as the head of S_v */
return FST;
```

V. Numerical Results

In this section, system level simulations are carried out. Referring to [12] and [9], we construct a LTE-A D2D network simulator. The parameters are given in Table I. As shown in Fig. 5, the performance results are presented for different network sizes. Each case is evaluated in average convergence time within 1000 trials (randomly generated topologies with fixed nodes). Note that the convergence time here is the measure of how fast the devices reach the state of complete synchronization. The average convergence time is the mean of the 1000 trials. Since currently there is no other related work which is under the framework of LTE-A, the compared

Algorithm 2: Head-Connect(v, S_v, FST)

```
if v has no adjacent vertex u ∉ S_v then
    return false;
while true do
    /*{u, v} is the maximum edge∉ S_v adjacent to v */
    while φ_v ≠ 1 do
        if receive PS_H from u then
            send PS_H;
            if receive PS_H from u then
                return true;
            else
                return false;
    send PS_H;
    if receive PS_H from u then
        send PS_H;
        return true;
    else
        return false;
```

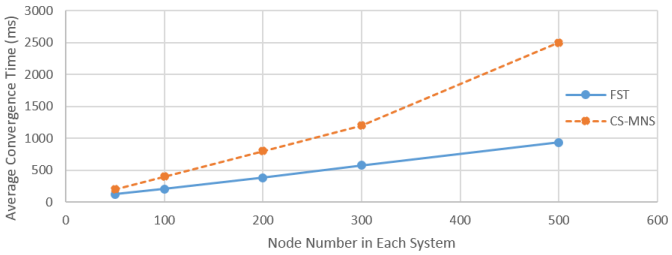


Fig. 5. The simulation results of applying FST and CS-MNS on networks with different scales. The test cases are randomly generated networks with 100, 200, 300, 500 devices. Each dot indicates the average convergent time within 1000 trials for each test case.

algorithms are chosen from other existing wireless communication systems [10]. Obviously, the centralized algorithms are invalid to our system. In addition, when we tried to implement the decentralized algorithms on our simulator, we found that most of them are either incompatible to the LTE-A system or unable to converge in random topologies. After thorough investigation, we only found the algorithm of clock-sampling mutual network synchronization (CS-MNS) [13]. As we can see in Fig. 5, the proposed FST outperforms CS-MNS in any size network. Also, we can observe that FST is even more efficient in larger systems due to the linearithmic time complexity.

Another performance evaluation is the exchanging messages during the converging progress. In Fig. 6, we can see that the trade-off of fast convergence is the number of exchanging messages. However, we can see that the deal is quite worthwhile. While FST largely outperforms CS-MNS in convergence time, the discrepancy of the two algorithms are always subtle. In fact, when in large networks (e.g. the case of 500 nodes), FST even narrowly defeated CS-MNS in terms of number of messages exchanged.

VI. Conclusion

In this paper, we proposed a distributed mechanism for D2D network ProSe, especially in LTE-A systems. The mechanism is able to achieve proximity discovery and synchronization at the same time, in both physical and application level. Some

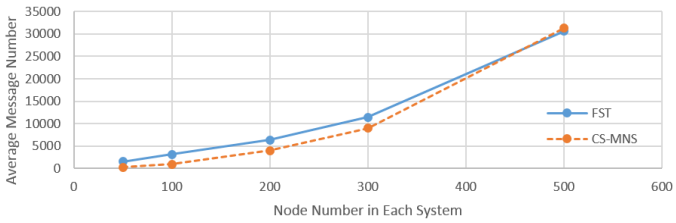


Fig. 6. The comparison in exchanging message numbers. Each dot indicates the average number of messages exchanged during the converging procedure within 1000 trials for each test case.

TABLE I
Simulation Parameter Values

Device Power	23 dBm (220 mW)
Threshold	-95 dBm ($3.16 * 10^{-10}$mW)
Device Density	50 devices in 100 m * 100 m areas
Propagation Model	Outdoor non-line-of-sight (dB) $P_{NLOS} = 43.5 + 25log_{10}(d)$ if $d < 60m$ $P_{NLOS} = 40.0 + 40log_{10}(d)$ otherwise
Shadowing Standard Deviation	10 dB
Time Slot	1 ms

essential operations can also be accomplished at the same time. In addition, a practical algorithm called FST is also given for resolving the problems caused by different topologies. Finally, the numerical results on LTE-A simulator show that the algorithm outperforms the other existing algorithms.

References

[1] Technical Report 22.803 v12.0.0, "Feasibility Study on Proximity Services (ProSe)," 3GPP, 2012
[2] Release-12 Working Item Description RP-122009, "Study on LTE Device to Device Proximity Services," 3GPP, 2012.
[3] R. E. Mirollo, S. H. Strogatz, "Synchronization of pulse-coupled biological oscillators," SIAP, vol.50, no.6, pp.1645-1662, 1990.
[4] G. Wener-Allen, G. Tewari, A. Patel, M. Welsh, R. Nagpal, "Firefly-inspired sensor network synchronicity with realistic radio effects," Sensys '05, pp.142-153, 2005.
[5] A. Tyrrell, G. Auer, and C. Bettstetter, "Emergent Slot Synchronization in Wireless Networks" IEEE Transactions on Mobile Computing, vol.9, no.5, pp.719-732, 2010.
[6] A. Tyrrell, G. Auer, and C. Bettstetter, "Fireflies as role models for synchronization in ad hoc networks," BIONETICS '06, pp.4, 2006.
[7] D. Lucarelli and I. J. Wang, "Decentralized synchronization protocols with nearest neighbor communication," SenSys '04, pp.62-68, 2004.
[8] K. Doppler, C. B. Ribeiro, and J. Kneckt, "Advances in D2D Communications: Energy efficient Service and Device Discovery Radio," Wireless VITAE '11, pp.1-6, 2011.
[9] C. Mehlfhrer, J. Colom Ikuno, M. Simko, S. Schwarz, M. Wrulich, and M. Rupp, "The Vienna LTE Simulators - Enabling Reproducibility in Wireless Communications Research" EURASIP Journal on Advances in Signal Processing, Vol. 2011, pp.1-13, 2011.
[10] J. Yick, B. Mukherjee, D. Ghosal, "Wireless sensor network survey," Computer Networks, Vol.52, Issue 12, pp.2292-2330, 2008.
[11] R. G. Gallager, P. A. Humblet and P. M. Spira, "A Distributed Algorithm for Minimum-Weight Spanning Trees," ACM Trans. Program. Lang. Syst. Vol.5, Issue 1, pp. 66-77, 1983.
[12] R1-130598, 3GPP TSG-RAN WG1 #72, "Channel models for D2D deployments," 3GPP, 2013.
[13] C. H. Rentel, and T. Kunz, "Clock-sampling mutual network synchronization for mobile multi-hop wireless ad hoc networks," MILCOM '07, pp.1,7, 2007.