Supplementary Material for the review of the manuscript:

D. Buranapanichkit and Y. Andreopoulos, "Distributed time-frequency division multiple access protocol for wireless sensor networks", *IEEE Wireless Communications Letters*, submitted.

The provided supplementary materials are:

- Model calculation in Matlab, provided in folder `/Matlab`. The code can be called via the Matlab command line via:

  `estimate_delay_until_balance_random_join(W,C);`

  where `W` is the total number of nodes (W_{tot} in our manuscript) and `C` is the total number of channels.
- NesC code for the implementation of the proposed TFDMA, provided in folder `/DesyncSwitch`. Please see below for instructions on installing TinyOS and compiling and running the provided code. The folder `/CC2420Radio` contains the radio interface that can be used with the provided code. The only modification performed in the default radio interface is the reduction of the backoff time to 1.2ms, in line with what is used by Degesys *et al* in the original DESYNC algorithm (ref. [4] of the manuscript). We tested with Crossbow imote2 nodes.

The documents are uploaded in the Supplementary materials area of the Manuscript Central (MC) system. The provided code and documents can be accessed via the single archive:

`supplementary_material.use_unzip_to_decompress`

if it is downloaded from the MC system and unzipped via a Zip decompressor program.

% TinyOS imote2 NesC code for desynchronization
% This code produces the multi-channel desynchronization of the manuscript:
% D. Buranapanichkit and Y. Andreopoulos, "Distributed Time-Frequency Division Multiple Access
% Protocol For Wireless Sensor Networks", *IEEE Wireless Communications Letters*, submitted.
% The provided code in folder `/DesyncSwitch` is including only the basic functionality of
% desynchronization. This was done for ease of illustration of the algorithms tested.
% The code is set for 8 channels; the total number channels can be easily selected within the source code.

Installing the Imote2 Development Environment for TinyOS 1.x

step 1 Installing Cygwin if you use Windows
- Download and install the latest version of Cygwin from Sourceware (http://sourceware.org/cygwin/)

step 2 Downloading the TinyOS1.x source code
- Install the latest version of TinyOS 1.x from SourceForge
- Configuring the TinyOS1.x Tree for Imote2, see http://shm.cs.uiuc.edu/files/docs/GettingStarted_AdvancedUsers.pdf (this document has complete instructions for all the steps required before you can use the provided code for desynchronization)

step 3 Installing the NesC compiler
- Download the latest version of the compiler from SourceForge

step 4 Installing the Wasabi tool suite
- Download the wasabi tool suite from the Intel website (http://www.intel.com/design/intelxscale/dev_tools/031121/)

Testing the application (instructions refer to Windows and Cygwin, straightforward modifications are required for usage under Linux)

step 1 Create the new folder for the application at directory `c:\tinyos\cygwin\opt\apps`
step 2 Update the interface file of the TinyOS 1.x for the Imote2
- CC2420RadioM.nc file at directory `c:\tinyos\cygwin\opt\tos\lib\CC2420Radio`

For the number of Imotes you would like to test with, repeat the steps below before using them to measure.
step 3 Compile the application for the Imote2 in the Cygwin shell:

```
cd $TOSROOT/apps/Desync
ADDRESS=x make imote2
```

with `x` being the node id

step 4 Plug the USB cable to the PC and directly to the Crossbow Imote2 (http://www.xbow.com). Turn Imote2 on.
step 5 Download the Imote2 application with the USB loader:

```
USBLoaderHost -p build/imote2/main.bin.out
```

Once all Imotes have been loaded with the application, you can also configure another 8 Imotes as base stations to record all messages in the 8 channels used. Switching on the base stations and the test Imotes produces the results of the manuscript.