Proceedings of the of the 17th Conference on Computer Science and Intelligence Systems pp. 601–604
DOI: 10.15439/2022F119
ISSN 2300-5963 ACSIS, Vol. 30

# Wireless Agent-based Distributed Sensor Tuple Spaces using Bluetooth and IP Broadcasting

Stefan Bosse
University of Bremen, Dept. of Computer Science, and Institute for Digitization
28359 Bremen, Germany
Email: sbosse@uni-bremen.de

*Abstract*— **Most Internet-of-Things (IoT) devices and smart sensors are connected via the Internet using IP communication directly accessed by a server that collect sensor information periodically or event-based. The spatial context (the environment in which the sensor or devices is situated) is not reflected accurately by Internet connectivity, and which is additionally not everywhere available. In this work, smart devices communicate connectionless and ad-hoc by using low-energy Bluetooth broadcasting available in any smartphone and in most embedded computers. Bi-directional connectionless communication is established via the advertisements and scanning modes. The communication nodes can exchange data via functional tuples using a tuple space service on each node. Tuple space access is performed by simple evenat-based agents. The Bluetooth Low Energy Tuple Space (BeeTS) service enables opportunistic, ad-hoc and loosely coupled device communication with a spatial context.**

## Introduction and Overview

THE number of embedded systems grows exponentially. Ubiquitous and pervasive computing introduced visible and non-visible low-resource and mobile devices. Most Internet-of-Thing (IoT) devices and smart sensors are still connected via the Internet using IP communication that are accessed by a server that collect sensor information periodically or event-based. Internet access is not everywhere available. Additionally, the residential time of mobile devices can be short, not suitable for ad-hoc connection-based and negotiated communication. Finally, the spatial context (the environment in which the sensor or device is situated) is not considered (or inaccurately determined) by Internet connectivity. even new 5G technologies will not fully solve context-aware communication [1]. In this work, smart devices communicate connectionless and ad-hoc by using low-energy Bluetooth available in any Smartphone and in most embedded computers, e.g., the Raspberry PI or ESP32 devices. Bi-directional connectionless communication is established via the advertisement and scanning modes used in parallel. The communication nodes can exchange small data or functional tuples using a tuple space service. Mobile devices act as tuple carriers that can carry tuples between different locations. Additionally, UDP-based Intranet communication can be used to connect tuple spaces.

Tuple spaces are widely used for data storage for loosely coupled distributed data processing systems. The Bluetooth Low Energy Tuple Space (BeeTS) service is a lazy distributed tuple space server and client. BeeTS uses Bluetooth and UDP broadcasting for tuple space interaction and tuple exchange. BeeTS supports tuples with an arity up to 4.

A tuple space provides a spatial context, i.e., tuple space access (by mobile devices) is limited to nearby devices, well suited for mobile networks [2]. Distributed tuple spaces can be connected by routers using IP communication if available. The router composes global space sets by tuple exchange and replication using hybrid rule- and event-based agents. These rules can be changed at run-time and the code can use Machine Learning algorithms to optimally distribute tuples under resource and spatial constraints.

The novelty of this work is two-folded. Firstly, an ubiquitous radio broadcast medium is used for low-distance communication in ad-hoc mobile networks combined with a unified tuple space paradigm. Secondly, the tuple space communication is performed by simple reactive event-based agents programmed in JavaScript that can be sent to a node via the tuple space operations, too. Agent-based message routing [3] is well suited in highly dynamic and unstructured network environments. The generative tuple space paradigm is well suited for ad-hoc mobile networks [4], especially if this paradigm is coupled with the agent paradigm [5].

## Communication

It is assumed that there is a broadcast medium $B$, e.g., using radio waves, which can reach a number of nodes $N_B=\{n_i\}_{i=1}^{k}$ defining a receive area/range coverage $C(B,N)(t)$ that changes over time $t$. $B$ can send broadcast messages $m$ to all listening nodes reachable by $B$. The set of nodes within $B$ can vary on time and spatial scale. Furthermore, it is assumed that there is a probability $p_i(m,r_{i,j},[t_0,t_l]) \in [0,1]$ that a message $m$ is received by a node $i$ sent by node $j$ in distance $r$ within a time interval $[t_0,t_l]$. These two assumptions are fundamental for the proposed distributed tuple spaces.

It is assumed that single packets that can be send over $B$ are strictly limited by a small number of bits, e.g., 200-300 Bits. This requires a compact and optimized message format, discussed in the next sub-section.

There are seven different message types:

- *OUT* stores a tuple in all tuple spaces receiving this message;
- *RD* and *INP* requesting tuples from all receiving tuple spaces;
- *TEST* checks for the existence of a tuple or set of tuples;
- *TUPLE* is either an initial message sending this tuple to all receiving nodes without; storing the tuple in the respective tuple space, or a reply to a tuple request;
- *IAMHERE* and *WHERE* messages are used for node search.

The message format consists of a message header and the data payload. The sequence number is required to detect the reception of multiple copies of the same message, a prerequisite for deployment with the Bluetooth device back-end that sends a message multiple times. The signature byte specifies the following tuple data pay-load. Depending on the back-end communication device and the supported packet format, the number of pay-load bytes can be very small. The signature field specifies the type of each tuple element with a tuple limit of four elements. For Bluetooth advertisement packages there is $N_{\mathrm{BLE}}$=32, for the UDP back-end it is at most $N_{\mathrm{UDP}}\geq$512. The message header and the data payload is encoded in an BLE advertisement packet using one device local name attribute (ASCII85 encoded) and seven 16 Bit service UUID attribute fields.

In contrast to typical tuple space services, the tuple operations are not atomic here. They can be executed at any given time point $t$ in the near future or never, and the set of reachable tuple spaces that execute the request is not bound and can be zero. There is no assumption that neither a message arrives at a specific node nor that request is processed successfully. There are filter rules processed by agents that can be prevent tuple operation execution, too. That means, the *INP* operation is only a suggestion to all receiving tuple spaces to remove a matching tuple. All operations pose a probabilistic behaviour, i.e., there is a probability $\geq 0$ that a message is processed.

The encoding of tuples is done automatically. Before a tuple is encoded and packed, a signature is derived, numbers are classified either in integer 16 Bit or float 32 Bit values depending on the actual value.

Devices can access remote tuple spaces of nearby neighbouring nodes (typically in the range of 1-10m) by using BLE broadcasting (called ble-ts). A device in advertisement mode will send out periodically advertisement message that contain a small payload depending on the advertisement message class. In this work, the pay-load is limited to 32 Bytes. There are 40 RF channels in BLE, each separated by 2 MHz (centre-to-centre). Three of these channels are called the primary advertising channels (labelled 37, 38, and 39), while the remaining 37 channels are called the secondary advertisement channels (they are also the ones used for data transfer during a connection). The primary channels are switched randomly in periods. On the other side, the scanning devices has to switch the (primary) receiving channels randomly, too. There is a probability $p$, that an advertisement packet is received if both scanner and advertiser are switch on the same channel and if there is no other sending within the receiving range creating collision (invalidation of the message).

In addition to the BLE broadcast communication, nodes that are connected to a local IP network can exchange tuple requests via UDP broadcast messages (called udp-ts). Although, UDP messaging is theoretically reliable, UDP broadcasting via wireless connections is not supported. Security is provided by a symmetric two-way encryption with format-preserving encryption of tuple messages using byte look-up tables.

## BEETS

The principle network architecture combining Bluetooth and UDP-IP broadcast communication technologies is shown in Fig. 1. Tuple messages can either be sent via Bluetooth advertisement (based on [7]) or via single UDP packets within a local IP network. BeeTS is programmed entirely in JavaScript and can be executed by node.js with a Bluetooth socket modules for BLE access, the *noble* module for the central BLE part, and *bleno* for the peripheral part. Note that BeeTS uses the peripheral and central (master) mode simultaneously (advertising and scanning), requiring a Bluetooth device with version $\geq$ 4.0. BeeTS is basically a small library module written in JavaScript. Smartphones act as mobile devices and provide both a rich set of sensors and BLE connectivity. Each communication back-end can receive tuple requests. If there is a listener installed for tuples (with pattern matching), incoming tuples (*TUPLE* message) can be consumed by the listener or not. Otherwise, incoming tuples are stored in the local tuple space.

There are agents acting as a bridge between the communication back-end and the tuple space. They can filter incoming messages and decide to reply immediately, to access the tuple space, or to discard the message. Agents are functional code that listen to incoming tuple requests. There can be more than one agent. Communication between agents is established via the tuple space, too.

A broadcast message sending via BLE enables the advertisement mode of the device for a specific time interval [$t$s,$t$e], shown in Fig. 1 (a). The duration of the time interval $\Delta t$ determines the receiving probability, the collision probability (if more than one station is sending), the number of advertisement packets that contain the message m, and the number of different messages that can be sent per second. The interval time $\Delta$t must at least 3 $\times$ $t$sw, with $t$sw as the

average channel switching time of the sender (and receiver). It is assumed that the sender and receiver have the same switching time, typically 100 ms. Important to note that channel switching introduces small dead time intervals (about 1-10ms). A suitable value for $\Delta t$ is about 500ms.

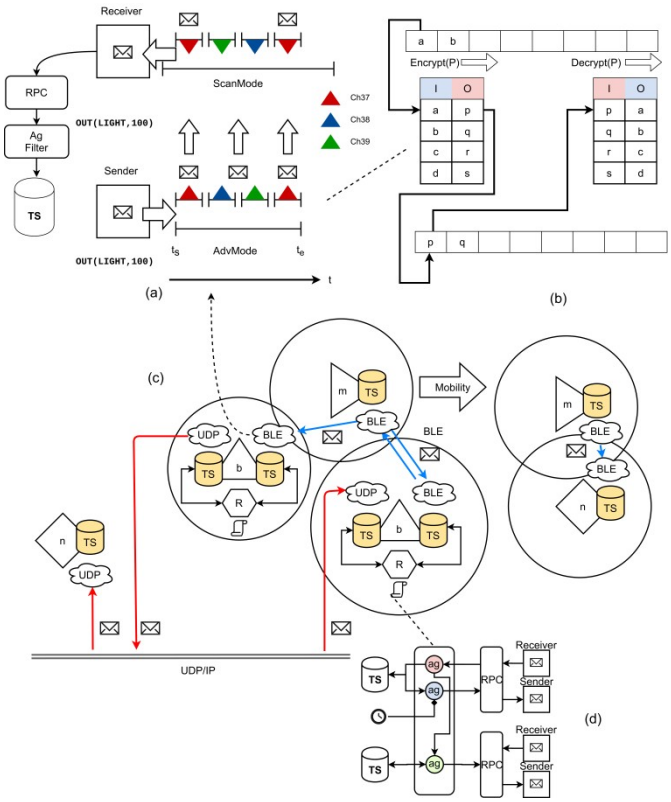


*Fig. 1. (a) BLE communication (b) Security by FPE (c) The hybrid network architecture using BLE and UDP-IP broadcast communication; n: stationary node, b: Stationary beacon, m: Mobile node, TS: Tuple space, R: Rule-based router (d) Message routing by agents between different TS*

Each physical communication interface (BLE/UDP) is attached to itws own tuple space, i.e., there ate two distributed space sets connected via BLE and UDP, respectively. This division is grounded in the spatial context of tuple spaces. Using BLE communication only nearby nodes can insert or remove tuples, whereas UDP communication enables tuple exchange over short and large distances, too. Tuple exchange between BLE and UDP tuple spaces is provided by a customisable router, shown in Fig. 1 (c). Application-specific routing rules (functional code) provide transfers based on patterns and content of tuples. The rule set is dynamic and can be changed at run-time. The router extends the visibility and scope of tuples based on adaptive code. The code can use Machine Learning, e.g., reinforcement learning, to improve tuple space distribution. The routers connect local spaces and compose organised global spaces.

Each time a message is received it is passed to the Remote Procedure Call layer (RPC). Among the message data, the sender MAC ID, a time stamp, and the signal strength is added to the message.

The BeeTS framework basically provides a communication platform using radio communication like Bluetooth or WLAN. The communication bandwidth of various devices can be significantly limited (e.g., in the case of Bluetooth advertisement mode that can be only 2 messages/second). One main feature of BeeTS nodes is the capability to execute event-based reactive agents programmed in JavaScript that perform, e.g., filtering of incoming tuple space requests. An agent is functional data consisting of private body variables (including functional values) and event handlers that are activated by incoming messages, sensors (if the host platform provides them), periodically, or only one time. Agents are executed in sand-boxed containers and are used for message filtering and routing between different tuple spaces, shown in Fig. 1 (d). The format-preserving encryption of tuple messages using look-up tables is shown in Fig. 1 (b).

## USE-CASE: DISTRIBUTED SMART BUILDING CONTROL

This use-case deploys three different node classes implementing a distributed building light control system:

1. Stationary beacons (Raspberry 3) equipped with BLE and WLAN connectivity and supporting ble- and udp-connected tuple spaces in both test and production deployment;
2. Mobile devices (battery powered RP Zero stacked with a smartphone for testing, stand-alone smart-phone in production systems) supporting ble- and udp-connected tuple spaces in test and ble-connected tuple spaces only in production environments;
3. A central monitoring and light control service supporting udp-connected tuple spaces.

Each node deploys at least one event-based agent that implements necessary node operations like interaction with mobile devices or users, and tuple filtering and bridging. Beacons consume and aggregate mainly sensor data from mobile (sensorised) devices like smartphones and forward micro-surveys from the central server to mobile devices. But beacons can initiate and manage micro-surveys, too. To minimise the number of sent tuples via the BLE device, the mobile nodes monitor the user behaviour by analysing the accelerometer and gyroscope sensors. Updates of light sensor tuples are only sent if either the light conditions changes or the mobile device was moved in space. For rapid prototyping, smartphones are using generic Web browser loading an application page from the locally attached Raspberry PI zero bundled with the smartphone. All sensor data is sent to the embedded computer that executes the mobile application logic and that performed the BLE communication.

Mobile devices uses their light sensor in conjunction with accelerometer and gyroscope sensors to estimate the ambient

light conditions and the user mobility by classifying the user activity in rest, smartphone use, and movement phases.

The measured light sensor data is processed by a sensor agent that tries to estimate if the smartphone is currently exposed to external light or if it is stored in a box. If external light is detected, sensor light tuples are sent via BLE. Nearby beacons distributed in the building about every 10-20m (and one per room/floor) collect these tuples and send aggregated sensor light values to the central server via udp-connected tuple spaces.

Among sensor tuples, there are micro-survey tuples that are sent from a beacon (initially delivered by the central server via the UDP tuple space) to mobile devices. If a device supports HMI (e.g., a smartphone), a short question is posted to a chat dialogue platform embedded either. The user can answer the question and the answer is passed back to the beacon (or any other beacon due to movement). The beacons collect the micro-survey replies and forward them to the central server.

For the evaluation of the loss rate of BLE tuple space communication, a partial set-up was chosen with four beacons at four different spatial positions and four mobile devices here all at the same position. An average loss below 10% can be achieved within a radius of about 5m. Some nodes can communicate over larger distances up to 10m. The tuple message send time interval has no significant impact on the loss rate within time interval [500s,2000s] and with this (small) set-up. If the number of nodes within the radio range increases, the loss rate will increase.

## CONCLUSION

In this work, The Bluetooth Low Energy Tuple Space (BeeTS) service enables opportunistic, ad-hoc and loosely coupled device communication with distributed tuple spaces that are used to exchange data between devices providing a spatial context with respect to data and communication. Smart devices access the tuple spaces by tuple message communication using event-based agents. The communication is connectionless and ad-hoc by using low-energy Bluetooth broadcasting available in any Smartphone and in most embedded computers, e.g., the Raspberry PI devices. Bi-directional connectionless communication is established via the advertisement and scanning modes used in parallel by transferring encoded tuple messages. Mobile devices act as tuple carriers that can carry tuples between different locations. Additionally, UDP-based Intranet communication can be used to connect tuple spaces. with spatial context. Multiple independent tuple spaces can be serviced on one network node bridged by agents. Among the tuple spaces, BeeTS implements simple reactive event-based agents. These agents perform tuple space management, interaction between devices and users, and they act as tuple filters and forwarders between multiple tuple spaces. A preliminary study showed the suitability of the broadcast communication for distributed ad-hoc networks preserving a spatial context lacking in other approaches. Analysis showed a low loss of BLE broadcast messages (about 10-20%) but higher and unpredictable loss rates of UDP broadcast communication using WLAN, even if N:1 unicast communication was used to simulate broadcast messages.

## APPENDIX: EXAMPLE AGENT

The following example shows an event-based agent programmed in JavaScript reacting on tuple messages. It consists of body variables and the event section. A specific tuple space can be selected.

```
var agent = {
  x  : 0,
  y  : 0,
  ..
  on : {
    'ts.udp:(TIME,?)':  function (ev) {
      ts.ble.notify(ev.tuple);
      return consumed?;
    },
    'ts.ble:(SENSOR,?,?,?)':function (ev) {
      log(ev.tuple,ev.from,ev.rssi)
      ts.udp.out(['EVENT',
                  JSON.stringify(ev.tuple),
                  ev.from,ev.time]);
      return consumed?;
    },
    init : function () {
      this.x=random(1,1000);
    },
    1000 : function (counter) {
      // called each 1000 ms
      return true
    },
    'sensor.light:abs(sensor-sensor0)>50':
      function (ev) {
        if (ev.sensor>500)
          ts.ble.notify(['ALARM',
                         'LIGHT','HIGH']);
      }
  },
}
Agent.create(agent);
```

## REFERENCES


[1] A. Gupta and R. K. Jha, “A Survey of 5G Network: Architecture and Emerging Technologies”, IEEE Access, vol. 3, 2016.

[2] P. Costa, L. Mottola, A. L. Murphy, G. P. Picco. "Teenylime: transiently shared tuple space middleware for wireless sensor networks." In Proceedings of the international workshop on Middleware for sensor networks, pp. 43-48. 2006

[3] E. Shakshuki, H. Malik, and X. Xing, “Agent-Based Routing for Wireless Sensor Network”, Lecture Notes in Computer Science,Advanced Intelligent Computing Theories and Applications. With Aspects of Theoretical and Methodological Issues, vol. 4681, pp. 68-79, 2007.

[4] N. Davies, A. Friday, S. P. Wade, G. S. Blair, “L2imbo: A distributed systems platform for mobile computing”, Mobile Networks and Applications 3(2), 143-156., 1998

[5] S. Bosse, “Unified Distributed Sensor and Environmental Information Processing with Multi-Agent Systems: Models, Platforms, and Technological Aspects”, ISBN 9783746752228, epubli, 2018

[6] S. Bosse, “BeeTS: Smart Distributed Sensor Tuple Spaces combined with Agents using Bluetooth and IP Broadcasting”, CoRR abs/2204.02464, 2022

[7] M. Nikodem and M. Bawiec, “Experimental Evaluation of Advertisement-Based Bluetooth Low Energy Communication”, Sensors, vol. 20, no. 107, 2020