Virtualising Testbeds to Support Large-Scale Reconfigurable Experimental Facilities

Tobias Baumgartner[1], Ioannis Chatzigiannakis[2,3], Maick Danckwardt[4], Christos Koninis[2,3], Alexander Kröller[1], Georgios Mylonas[2,3], Dennis Pfisterer[4], and Barry Porter[5]

[1] Dept. of Computer Science, Braunschweig University of Technology, Germany
{tbaum,kroeller}@ibr.cs.tu-bs.de
[2] Research Academic Computer Technology Institute, Patras, Greece
{ichatz,koninis,mylonasg}@cti.gr
[3] Computer Engineering and Informatics Department, University of Patras, Greece
[4] Institute of Telematics, University of Lübeck, Germany
{danckwardt,pfisterer}@itm.uni-luebeck.de
[5] Computing Department, Lancaster University, UK
b.porter@lancaster.ac.uk

Abstract. Experimentally driven research for wireless sensor networks is invaluable to provide benchmarking and comparison of new ideas. An increasingly common tool in support of this is a testbed composed of real hardware devices which increases the realism of evaluation. However, due to hardware costs the size and heterogeneity of these testbeds is usually limited. In addition, a testbed typically has a relatively static configuration in terms of its network topology and its software support infrastructure, which limits the utility of that testbed to specific case-studies. We propose a novel approach that can be used to (i) interconnect a large number of small testbeds to provide a federated testbed of very large size, (ii) support the interconnection of heterogeneous hardware into a single testbed, and (iii) virtualise the physical testbed topology and thus minimise the need to relocate devices. We present the most important design issues of our approach and evaluate its performance. Our results indicate that testbed virtualisation can be achieved with high efficiency and without hindering the realism of experiments.

1 Introduction

Experimentally driven research for wireless sensor networks has been instrumental in advancing the state of the art in recent years; new sensing applications, network architectures and protocol stacks have been optimised to operate over varied radio technologies, restricted resources and specific deployment strategies. The most commonly applied technique is *simulation* which allows rapid development, offers debugging tools and enables easy repeatability. A natural step beyond this is to implement the system on real hardware platforms and perform experiments in controlled testbed environments. This allows researchers

J. Sá Silva, B. Krishnamachari, and F. Boavida (Eds.): EWSN 2010, LNCS 5970, pp. 210–223, 2010.

to escape the inherent limitations of simulation regarding the available hardware characteristics (e.g. buffer sizes, available interrupts) and communication technology behaviour (e.g. transmission rates, interference patterns).

In the majority of cases, due to the costs of hardware, researchers evaluate their solutions in local testbeds of limited size. While small testbeds provide useful insights into the effectiveness of the system in real conditions, they tend to offer limited support in terms of heterogeneity, scalability and mobility. Furthermore, in most cases, a tightly coupled network and software architecture is followed on a testbed, thus limiting the number of possible configurations of that testbed.

In order to overcome limitations in scale, a number of testbeds of significant size have been developed in the last few years. Their size currently ranges up to 1000 nodes, and there is a trend towards building *even larger* testbeds as seen by projects such as WISEBED [14] and SENSEI [10]. This trend continues to serve more accurate experimentation – and therefore high quality research – in realistically-sized networks towards the scales imagined by the initial vision of sensor networking that dealt with using thousands or even tens of thousands of nodes.

Given this clear and continuing need for large open testbeds in WSN research, certain critical questions are posed: i) how do we deal with the ever-increasing total-number-of-nodes demand, ii) how do we combine large testbeds with heterogeneity (in available sensors, radios, computational resources, etc.), iii) how can we maintain a very large WSN testbed efficiently? Furthermore, how can we cater for hybrid simulation approaches, i.e., the combination of real and simulated testbeds in order to produce extremely large-scale WSN testbeds? Moreover, how do we utilize the facilities provided by these testbeds and adapt them to each experiment's needs; i.e. how can we define and use specific network topologies that fit into our target application domain?

We argue here that an efficient and flexible answer to such problems is the use of *federated testbeds* that unite isolated WSN testbed "islands" with the use of a *virtual links* concept. We propose the use of virtualised network links in the following ways:

- *Between physically distinct testbeds* of varying features (location, size, etc.) as a whole, but also *between specific nodes* of such testbeds, resulting in larger testbeds with customised cross-network edges,
- *Between nodes inside a single testbed*, thus defining a customised network topology,
- *Between real and simulated nodes*, enabling hybrid simulation for massive network sizes.

A virtual link essentially enables two testbed nodes, that have otherwise no direct physical radio connection, to communicate in a way that is transparent to the user applications; additionally, existing 'links' (i.e. reachability within one-hop radio range) can be selectively deactivated between neighbouring nodes. Both kinds of virtualisation are done in a way that is entirely transparent to a deployed application.

The major challenge arising from using such an approach is in the extent to which this virtualisation affects the realism of the experiments conducted – the tradeoff between the ability to extensively scale and reconfigure testbeds in a straightforward way and its impact on the realism of results. In relation to this we currently target only experiments which use higher layers of network abstraction, avoiding those which operate at the the MAC layer. However, we believe that the "simulation" of network links and the resulting federated testbeds will prove itself largely beneficial to the research community.

In this work, we provide a systematic definition of our testbed virtualisation concepts, discussing in-depth design, architecture and implementation issues of our approach. We provide an evaluation of our work to demonstrate the feasibility of testbed virtualisation, comparing results from real network topologies against virtualised ones. Our results show that in many cases virtualisation can be efficiently integrated into testbeds without having a profound effect on the experimental results' realism.

This paper is structured as follows: a description of related work follows in Section 2. An in-depth discussion of our virtual link service follows in Section 3, with a set of experiments and results described in Section 4. This is followed by an example application (i.e., an experiment using virtual links) in Section 5. A discussion of our results is provided in Section 6.

2 Related Work

There is a significant body of existing work in the area of sensor network simulation and testbed infrastructures, with a number of works following a hybrid approach in the last few years; charasteristic examples of this approach are [7,8,3,12]. In such approaches part of the experiment is conducted in simulation and part on real hardware, with the ratio varying in different approaches. In some cases, only the wireless communication channel of the real devices is utilised with the rest of the software being executed inside a simulator. In other cases the software is executed iteratively on real and simulated devices with certain arbitration and timing schedules applied. These concepts are somewhat related to our own, but we aim at using virtual links between real and simulated devices *in real time and simultaneously*.

There is also significant work regarding WSN testbeds and their respective management and debugging software. Large testbeds such as Trio [2], Motelab [13], TWIST [4] or SIGNETLAB [1] are accompanied by software that provides users with the facilities to conduct experiments with the testbed nodes, but are generally limited in their adaptability and configurability to the users' needs. Capabilities such as reconfiguring the network topology, or federating multiple testbeds to form a larger virtualised facility are to our knowlegde not provided in these cases.

Virtualised network links or federated testbeds and their use in network testbeds are in themselves not a new concept, with projects such as Planetlab implementing similar concepts. Additionally one recent approach dealing

with similar issues is [5], where the infrastructure and software of the Kansei testbed is combined with that of GENI in order to provide a unifying solution for discrete testbeds. Also, the Senslab project [11] aims to unify 4 discrete heterogeneous testbeds into a single one of 1000 nodes. However, all of these works lack a generalised approach allowing arbitrary configurations between real and virtual. Overall, our approach aims at providing a unifying abstraction for discrete testbeds and increased flexibility in defining network topologies, without hindering the efficiency and realism of using a testbed versus pure simulation.

3 Virtual Links and Federated Testbeds

In this section we describe in detail our approach to virtualising testbeds. We define a virtualised testbed as either a single physical testbed with a virtualised topology; two or more physically distinct testbeds federated into a single unified testbed; a simulated testbed similarly federated with a physical testbed; or any combination of the above.

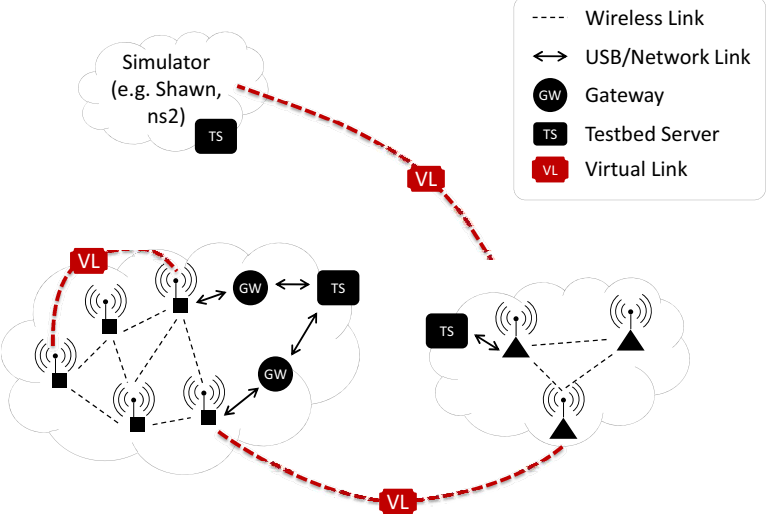


Fig. 1. The architecture of virtualised testbeds

The key components of our architecture are shown in figure 1. Each testbed – physical or simulated – is represented by a *testbed server* which acts as the Internet-facing gateway to the testbed. A testbed itself is composed of a number of *sensor nodes* which can communicate with the testbed server (potentially via gateway devices inside a physical testbed).

A *virtual link* is then a (unidirectional) connection between two nodes – in the same or in different testbeds – which would not normally be able to communicate.

An arbitrary number of virtual links can thus be created to define a virtualised topology and federate distinct testbeds. We can also *deactivate* existing physical reachability between two nodes by selectively dropping packets to allow complete topology control.

In more detail, virtual links are enabled with a special piece of software on each sensor node – a *virtual radio* – which contains a routing table of the form {ID, interface}, such that when sending a message to a specific node ID the radio can decide on which 'interface' to send this message; the node's real radio or the virtual interface which forwards the message to the testbed server (where it is routed onwards as appropriate).

In the remainder of this section we discuss the virtual radio software in detail and its interaction with a testbed server. Following this we describe the unified message format which allows heterogeneous nodes to communicate, we discuss the ways in which virtualised topology can support link quality modelling, and finally we describe how simulation is integrated into real-time testbed experiments.

3.1 Topology Virtualisation

Virtualising topology involves two key elements, *virtual radio* components, and *testbed servers*, shown in Figure 2.

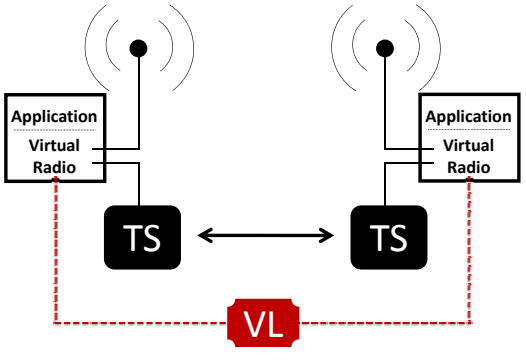


Fig. 2. The communication beween virtual radio drivers on sensor nodes

At the start of an experiment, the IDs of virtual radios across the entire virtualised topology are configured by an overall controlling component, ensuring uniqueness. Virtual topology itself is configured by each testbed server informing its local sensor nodes of their virtual neighbours, where a virtual neighbour entry in a sensor node's virtual radio simply consists of an ID along with 'virtual', meaning any packets to this ID should be sent to the testbed server for further routing. How messages reach the testbed server depends on the architecture of the deployed testbed – routing may be via an out of band backbone infrastructure when sensor nodes are connected 1:1 with gateway devices, or alternatively may

reuse the wireless medium of the sensor nodes in testbeds where not every sensor node is directly connected to a gateway device. In either case the procedure is transparent to the application software.

The process of sending a message thus works as follows: applications on a sensor node send a packet to its virtual radio component. On some operating systems (e.g., TinyOS), using a virtual radio instead of a real one is simply a matter of component configuration. On others, it may require changing radio function calls in the application's source. The virtual radio component then uses its local routing table (configured by the testbed server as above) to decide on which interface to send this message – via the real radio or the virtual topology service via the testbed server. When broadcasting, a packet is simply sent on both interfaces.

If the message is sent to the testbed server, the server examines the destination ID of the packet and forwards this either to another testbed server which is responsible for that node, or to the corresponding node in the local testbed. If the packet is broadcast, the testbed server forwards the message to all virtual neighbours of this node (i.e. generating one message for each neighbour).

Finally, when receiving a message on the *real* radio interface, the virtual radio component checks its routing table to determine whether or not the sender is configured to be a neighbour in the currently configured topology, and if not then the packet is simply dropped and so never reaches the application. All parts of this procedure can be completely transparent to the application, which can simply see a radio component conforming to a common radio API.

3.2 Message Format

Packets traveling over virtual links between testbed servers have a common format, and in cases where virtual links exist between nodes of different types, the local testbed server performs appropriate translation of the packet to a format suitable for use at the destination node. This process is also used when a single testbed has heterogeneous nodes – or nodes running different operating systems – with virtual links between them.

The common packet format therefore abstracts over different concepts of link quality between platforms (such as LQI or RSSI values) and other differences in the types of fields present in packets, different offsets for the same fields within packets, or different lengths of addresses. For this reason, we define a generic representation of a packet that is used when testbed servers forward messages from a virtual link between nodes. An example of a generic packet is shown below:

```
<?xml version="1.0" encoding="UTF-8">
<Node2Node_Packet>
      <sender_ID ID="urn:testbed1:node1" />
      <destination_ID ID="urn:testbed2:node2" isBroadcast="false" />
      <message>A1 DF 63 8B</message>
      <LQI>200</LQI>
      <RSSI>199</RSSI>
      <Options>ACK</Options>
</Node2Node_Packet>
```

This example contains the ID of the sender node (sender_ID), the ID of the destination node (destination_ID), whether the message is a broadcast, the payload of the message, an LQI value, an RSSI value and optional flags such as whether the sender needs an acknowledgment.

3.3 Modelling Link Characteristics

In addition to the basic routing functionality involved in virtualisation, testbed servers may perform traffic shaping on messages sent over virtualised links according to some desired model, for example emulating lossy channels or interference. Although beyond the scope of this paper, it is easy to plug such models into the software used at testbed servers.

3.4 Simulation Considerations

Connecting a simulator to a real testbed presents some unique challenges. Our motivation behind it is to enable ultra-large-scale experiments in which a large number of simulated nodes provide the macro-view of an experiment and serve as a test load to a relatively small number of real sensor nodes, on which the experiment outcome is measured. Using topology virtualisation the real nodes can be placed anywhere within the broader simulated topology.

Simulator integration follows the same implementation pattern as physical testbeds, i.e., the simulator is connected to a testbed gateway. A real-time enabled network simulator can be easily adapted to such an architecture. We chose the Shawn [6] network simulator for our experiments. The required modifications are also possible in other simulation environments.

Virtual link integration involves three major steps:

- Real-time simulation is required for a shared time basis between real and simulated nodes.
- Multi-threaded injection of messages into the simulation whenever a real sensor node (or alternatively a simulated node from another simulator) sends a message over a virtual link.
- Appropriate message 'routing' inside the simulator must be added, such that the simulator's testbed gateway routes messages sent from simulated to real nodes over its network connection.

This approach implies that the simulator is able to execute the simulation of nodes fast enough to keep up with real time; Shawn is a high-level simulator, and can easily do this for thousands of simulated nodes.

4 Evaluation

In this section we evaluate the *realism* of virtual links in experiments conducted over a federation of separate physical testbeds. Furthermore, we evaluate the

efficiency of our implementation in terms of *latency* (transmission delay of messages exchanged over virtual links) and *scalability* (increase in latency as the volume of messages over virtual links increases).

We implement the virtual radio component (see section 3.1) on three WSN operating systems: TinyOS, Contiki, and iSense. This makes our approach operable on a wide variety of hardware platforms such as the Crossbow mote series, Tmote Sky, ScatterWeb motes and iSense nodes. Our testbed server software is implemented in Java using Web Services for inter-server communication.

We evaluate the time to transmit a message over (i) the physical hardware radio, as a benchmark, (ii) a virtual radio using the UART to send virtual link messages to a directly connected gateway device (referred to as *Setup-I*), and (iii) a virtual radio implemented using the physical radio to forward virtual link messages to a gateway-connected sensor node (referred to as *Setup-II*).

We test the hardware radio in a simple two-node topology, providing a reference result for the speed of real radio messages. For the virtual radio tests we use two physically separate sensor nodes that are in different testbeds, in one case such that each sensor node is connected directly to a gateway device, and in a second case where each sensor node must use its hardware radio to forward virtual link messages to another sensor node which is directly connected to a gateway device – these represent the two most common kinds of WSN testbed deployments. In the virtual radio tests we have two testbed servers (one for each testbed) connected via gigabit Ethernet in the same LAN, thus simulating a full virtual link message transport procedure (as illustrated in figure 2).

For each type of link and each hardware platform considered we transmit a total of 1000 messages. Table 1 shows the minimum, maximum and average times taken for a message to be sent from an application at one sensor node and arrive at the other[1].

Table 1. The min/avg/max message transit times for 3 platforms using a physical radio, a virtual link over UART (Setup-I) and a virtual link over radio (Setup-II)

Hardware Platform	Physical Radio			Setup-I			Setup-II		
	min	avg	max	min	avg	max	min	avg	max
ScatterWeb	72.2ms	75ms	81.6ms	4.7ms	5ms	5.2ms	149.4ms	155.3ms	167.5ms
telosB	38.7ms	40.2ms	43.3ms	4.7ms	5ms	5.3ms	81.2ms	85.6ms	92.4ms
iSense	6.3ms	7ms	8.1ms	4.7ms	5ms	5.2ms	14.1ms	17.9ms	20.6ms

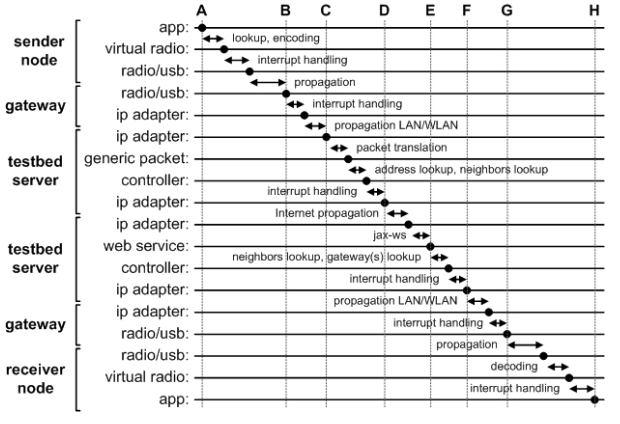
The results show significant variation in the message transit times when using the physical hardware radios of the different platforms; this is caused by differences in hardware design. In the iSense platform for example the JN5139 microcontroller integrates an on-chip CC2420 radio module thus increasing the

[1] To accurately measure the transmission time over the ScatterWeb platform we used a high-precision external clock, as the internal hardware clock has low accuracy.

speed of communication. By contrast, on the TelosB platform, the MSP430 controller has an external CC2420 radio module accessed via an SPI bus, and the ScatterWeb node has an 868Mhz radio with a 19.2 kbit/s data rate.

When we use virtual radio links sending messages via UART on the other hand we see that all platforms perform almost identically. This is because the UART modules used by the different hardware platforms conform to the same serial protocol and apply the same settings (115200 bps 8N1). When using virtual radio links sending messages via the physical radio modules, we of course see similar variations in transit times caused by the hardware differences noted above.

These results demonstrate that the transmission times using virtual radios over UART are in fact faster than using physical hardware radios for all hardware platforms considered, even though this involves transport through the testbed servers. This allows us to potentially do further processing on virtual radio packets (such as modelling link characteristics) before delivering them to the final device without impacting on realism, or to simply impose a delay on such packets to make them comparable to physical radio message transit times. Alternatively we could send virtual radio messages between testbeds over the Internet and still deliver them in reasonable time.



Period	Setup-I	Setup-II
A-B	2.06ms	6.93ms
B-C	0.4ms	0.4ms
C-D	0.01ms	0.01ms
D-E	0.14ms	0.12ms
E-F	0.01ms	0.01ms
F-G	0.39ms	0.4ms
G-H	2.05ms	6.93ms
Total	4.96ms	14.69ms

Fig. 3. The timing of a transmission for each layer invoked of the sender, receiver and intermediate nodes for two different physical testbed setups

To further explore the sources of delay in sending messages over virtual radio links we instrumented as many steps as possible in each stage of the transport procedure, shown in Figure 3, from the application on the sending sensor node to the gateway device (labelled stage A-B), processing inside the gateway device (B-C) and testbed server (C-D), and the reverse of this for the second testbed. We use the same two virtual radio link implementations as before; Setup-I being over UART and Setup-II over virtual link message transported via the physical radio to a gateway. The average delays were again calculated based on a total

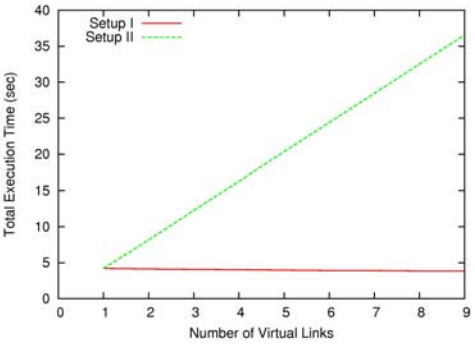


Fig. 4. Execution time for different number of virtual links on different testbed setups

of 1000 message exchanges for each setup considered. These results only show the iSense platform, since the periods A-B and G-H can otherwise be calculated from Table 1 for the other hardware platforms.

We conclude our performance evaluation by examining the scalability of our implementation. To do this we use a single testbed only with 10 sensor nodes. Nodes 1–9 are in one hop physical radio range of node 0 (the sink), to which they each transmit 1000 messages. We begin by using real hardware radio transmission for all nodes, then replace each of nodes 1–9's links to node 0 with virtual radio links. In each case we measure the total execution time for the sink to receive all 9000 messages. The results of this experiment are shown in Figure 4. The results indicate that testbed setup I, using UART for all virtual radio messages, is capable of delivering all 9000 messages over the virual links with almost no delay when compared to physical links. On the other hand, the total execution time of the experiment under testbed setup II, using the hardware radio to transport virtual radio messages via a single gateway device, linearly increases with the total number of messages transmitted over the virtual links. This is because in the latter case the single gateway node creates a bottleneck, and thus the total execution time increases with the number of messages transmitted over the virtual link (this is a worst-case scenario where only one gateway node exists).

5 Example Experiments Using Virtual Links

Having demonstrated the raw performance of our virtual radio implementations we now examine two characteristic WSN applications: sensor data aggregation and detection of a network partition. We execute these on testbeds using a mixture of real and virtualised topology, aiming to show that our approach does not impact the application's view of the network's behaviour – and therefore does not negatively affect results of experiments with these applications.

We use 3 different configurations: (i) a single testbed without any virtual links, (ii) two testbeds federated using virtual links whose testbed servers are on

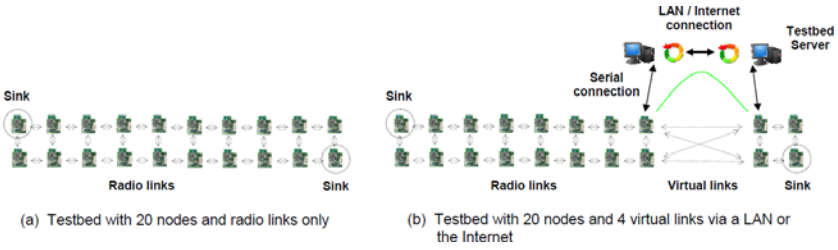


Fig. 5. Testbed Configuration

the same LAN, and (iii) two testbeds federated with virtual links whose testbed servers are at different locations within the same country and communicate over the Internet. Each configuration uses the same total number of nodes in the same topology, but in the virtualised cases some nodes are in different testbeds. For each configuration we evaluate how the applications perform to assess the effect of virtual links.

Sensor Data Aggregation. In this classic application, each node maintains an aggregate value (e.g. average) of a sensor reading (e.g. temperature) from all its neighboring nodes. Periodically all nodes report these aggregated values to the control center of the network (i.e. the sink). Each node broadcasts its sensor reading every 5sec. Nodes collect all data received and maintain the average value. Every 50sec each node sends the value to the sink using a simple flooding algorithm.

We deploy a single testbed without any virtual links (configuration I) that consists of 20 nodes arranged in two parallel lines of equal size with 1 meter distance between each node. The power output of radio interfaces is configured to achieve a maximum communication range of $1.5 - 2$m. The sink is placed at the top-left corner of the network (see Fig. 5a).

For the experiments with virtual links (configurations II and III), the testbed is separated in two parts consisting of 16 nodes and 4 nodes (Figure 5b). All nodes are directly connected to gateway devices via USB links (i.e. virtual radios using UART). We configure four virtual links between the border nodes of the two separated testbeds. In configuration II the two testbeds are within the same university LAN and are connected via a 100mbit Ethernet backbone. In configuration III the two testbeds are at different universities in the same country and are connected via the Internet. The connection between the two testbed servers in configuration III achieves an average ICMP echo request time of 12ms (over 100 echo requests) and a traceroute reported 15 hops.

For our evaluation we measure the average number of messages (i) sent by each node, (ii) received by each node via the actual radio component, (iii) received by each node via the virtual radio component and (iv) received by the sink for each 50sec cycle. We executed 10 experimental runs of 15min each for each testbed configuration. Table 2 shows the results for each metric over the three different testbed configurations when using iSense nodes (with similar results holding for

the other hardware platforms), demonstrating that the overall network behaviour in terms of messages received at the sink node is comparable in all cases, and general network behaviour in terms of messages sent and received is similar.

Table 2. Evaluation metrics for the Data Aggregation Example Experiment

Average number of messages	Configuration I	Configuration II	Configuration III
sent per node	373.53	377.95	386.95
received per node via real radio	1667.05	1085.47	954.95
received per node via virtual radio	–	771.25	806.75
received by sink per cycle	13.11	12.88	13.61

In performing these experiments, due to the speed of virtual links over real ones, it was necessary in configuration II to introduce a random delay of 1–3ms per virtual link message in order to match the behaviour of the physical radios; no such delay was imposed for configuration III, as the inherent delay of the national-level Internet link made the virtual links comparable to the real radio links. For comparison, using the ScatterWeb platform, with its slower hardware radio, we found random delays of 70–80ms (configuration II) and 50–65ms (configuration III) appropriate, while for the TelosB platform we used 32–40ms and 20–28ms respectively.

Partition Detection. Here we use the same network topology as in the previous application but position an additional sink at the other side of the network. In this application we wish to detect if the network is partitioned due to the failure of an intermediate node or due to a lossy radio link; in case of a partition we issue an alarm at the sink. To do this we implement the algorithm of [9], in which each sink essentially sends a beacon message every 5sec to the other sink by flooding it through the network. If a beacon message gets lost between the two sinks then an alarm message is generated.

For our evaluation we measured the average messages (i) sent by each node, (ii) received by each node via the real radio, (iii) received by each node via the virtual radio, (iv) sent by both sinks and (v) received by the sink for each 50sec cycle, with results shown in Table 3 (where partition alarms are occasionally raised due to lossy radio channels).

The results in Table 3 show that the network behaviour from the application's point of view (in terms of generated alarms) is comparable in both real and virtualised testbeds, and that network behaviour otherwise in terms of the numbers of messages sent and received is comparable.

Finally, table 4 shows how long it takes to send a partition beacon from one sink to the other and return to the initial sink. The roundtrip times here are almost identical for configurations I and II while for configuration III they are slightly higher due to latency introduced by the Internet link.

Table 3. Evaluation metrics for the Partition Detection Example Experiment

Average number of messages	Configuration I	Configuration II	Configuration III
sent per node	343.33	320.94	341.87
received per node via real radio	1395.45	870.45	857.05
received per node via virtual radio	–	611.51	687.76
sent in total by both sinks	355.12	361.60	360.32
Generated alarms (both sinks)	19.33	20.25	18.71

Table 4. Roundtrip times for one beacon message

	Configuration I	Configuration II	Configuration III
Minumum [ms]	46.66	42.24	49.39
Average [ms]	75.09	73.15	88.73
Maximum [ms]	110.99	100.51	139.22

6 Conclusion

Virtual network links, as defined and used in this work, are a means of easily reconfiguring and federating testbeds into large-scale networks with direct control over the topology. We have demonstrated that the approach is both realistic and performant enough for experiments at the higher network layers such that applications cannot distinguish between a fully real topology and a partially virtualised one. While we expect that experiments with MAC layer algorithms may reveal more subtle artifacts of virtualisation, and so would need care in deriving results, we believe that our approach is nonetheless highly beneficial in enhancing the utility of a single testbed beyond its fixed physical topology, in federating testbeds to enable extremely large scale experiments on real hardware and in enabling the integration of simulation for even larger networks.

When building federated testbeds for scale, we have shown that the interconnection of two closely located testbeds, i.e. with short delays between testbed servers, works very well in practice – as the virtual links operate significantly faster than physical links, an experiment can be tuned so that applications cannot detect that they are running in a physically separated network.

When moving to wider-area networks, we are limited by the existence of a sufficiently fast Internet backbone. While connections over 15 hops in a university-connecting national network fulfill such a requirement, we acknowledge that latency may degrade and affect realism too much in large intercontinental federations. However, we believe that future high-speed broadband networks will make this issue diminish in time.

We conclude that topology virtualisation is a promising approach to create large federations of physically separated and heterogenous networks. Building nationwide testbeds is achievable with today's technology, and we believe that worldwide networks are viable in the near future.

Acknowledgments

This work has been partially supported by the European Union under contract numbers IST-2008-224460 (WISEBED). The authors would like to thank Geoff Coulson, Sándor Fekete, Stefan Fischer, Qasim Mushtaq and Paul Spirakis for their insightful comments and ideas.

References

1. Crepaldi, R., Friso, S., Harris III, A.F., Zanella, A., Zorzi, M.: The design, deployment, and analysis of signetLab: a sensor network testbed and interactive management tool. In: WiNTECH 2006: Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization, pp. 93–94. ACM, New York (2006)
2. Dutta, P., Hui, J., Jeong, J., Kim, S., Sharp, C., Taneja, J., Tolle, G., Whitehouse, K., Culler, D.: Trio: enabling sustainable and scalable outdoor wireless sensor network deployments. In: Proceedings of the 5th international conference on Information processing in sensor networks (IPSN 2006), pp. 407–415. ACM, New York (2006)
3. Girod, L., Ramanathan, N., Elson, J., Stathopoulos, T., Lukac, M., Estrin, D.: Emstar: A software environment for developing and deploying heterogeneous sensor-actuator networks. ACM Trans. Sen. Netw. 3(3), 13 (2007)
4. Handziski, V., Köpke, A., Willig, A., Wolisz, A.: TWIST: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In: REALMAN 2006: Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality, pp. 63–70. ACM, New York (2006)
5. Wiki for the Kansei GENIE project, `http://sites.google.com/site/siefastgeni/`
6. Kröller, A., Pfisterer, D., Buschmann, C., Fekete, S.P., Fischer, S.: Shawn: A new approach to simulating wireless sensor networks. In: Design, Analysis, and Simulation of Distributed Systems 2005 (DASD 2005), April 2005, pp. 117–124 (2005)
7. Österlind, F., Dunkels, A., Voigt, T., Tsiftes, N., Eriksson, J., Finne, N.: Sensornet checkpointing: Enabling repeatability in testbeds and realism in simulations. In: Roedig, U., Sreenan, C.J. (eds.) EWSN 2009. LNCS, vol. 5432, pp. 343–357. Springer, Heidelberg (2009)
8. Park, S., Savvides, A., Srivastava, M.B.: Sensorsim: a simulation framework for sensor networks. In: MSWIM 2000: Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems, pp. 104–111. ACM Press, New York (2000)
9. Ritter, H., Winter, R., Schiller, J.: A partition detection system for mobile ad-hoc networks. In: Sensor and Ad Hoc Communications and Networks, IEEE SECON 2004, October 4-7, pp. 489–497 (2004)
10. SENSEI project website, `http://ict-sensei.org/`
11. Very large scale open wireless sensor network testbed, `http://www.senslab.info`
12. Wen, Y., Zhang, W., Wolski, R., Chohan, N.: Simulation-based augmented reality for sensor network development. In: SenSys 2007: Proceedings of the 5th international conference on Embedded networked sensor systems, pp. 275–288. ACM, New York (2007)
13. Werner-Allen, G., Swieskowski, P., Welsh, M.: Motelab: A wireless sensor network testbed. In: Fourth International Conference on Information Processing in Sensor Networks (IPSN 2005). IEEE, Piscataway (2005)
14. WISEBED project website, `http://www.wisebed.eu/`