An Overlay Network for Integration of WSNs in Federated Stream-Processing Environments

Niko Pollner*, Michael Daum*, Falko Dressler†, Klaus Meyer-Wegener*
*Department of Computer Science, University of Erlangen, Germany
†Institute of Computer Science, University of Innsbruck, Austria
{niko.pollner,md,kmw}@cs.fau.de, falko.dressler@uibk.ac.at

Abstract—**This paper presents an approach for seamless integration of hosts in heterogeneous networks in the context of data-stream processing. The integration of multiple heterogeneous hosts from the sensor-node level up to the level of high-performance workstations is one of the most promising concepts for extensive and efficient analysis of streaming data. For controlling such a network, communication between hosts is needed, e. g., to initiate stream processing, to configure queries, and to transmit streams. One of the key challenges is global addressing and transparent yet efficient data exchange despite diverse, differently capable networks involved.**

For this purpose, we developed a cross-platform overlay network that enables transparent communication between autonomous stream-processing systems on different hosts in miscellaneous networks for both data streams and control commands. The system directly uses underlying native protocols within each network so that the most efficient communication method is applied. Furthermore, global addressing of instances of stream-processing systems and routing over the individual communication paths is provided.

I. Introduction

In the scope of our *Resource-constrained Distributed Stream Processing* (RDSP) project[1], the *Data Stream Application Manager* (DSAM) [1] manages distributed stream processing queries that includes the reorganization of those queries. In this article, stream refers to a stream of tuples and not to a multimedia stream. DSAM integrates heterogeneous *Stream Processing Engines* (SPEs) and *Wireless Sensor Networks* (WSNs) and provides a federated stream-processing environment. In the remainder of this article, both full-fledged SPEs like Borealis [2] usually running on workstations and stream-processing WSN nodes are subsumed as *Stream Processing Systems* (SPSs) for reasons of simplicity. Queries can be submitted to the central DSAM controller running on a certain host. The queries are automatically partitioned, optimized, and disseminated to the SPEs available for processing. Section II details the architecture of DSAM.

For dissemination of partial queries, a communication path to each SPE and WSN node is needed. As long as all hosts reside in a common network this imposes no problems and standard protocols like TCP/IP can be used. But on sensor networks the TCP/IP protocol is often not natively available. Moreover, there exist many different connection techniques and protocols for communication between sensor nodes. The WSN's connection to the stationary network is usually achieved by connecting the gateway node and a stationary computer via a serial line that needs its own special communication protocols.

The DSAM controller needs to communicate with the different SPSs to start and stop stream processing, disseminate new queries and maintain running queries. Communication among the SPSs is necessary to forward streams between the SPSs involved in the streams' processing. To provide easy expandability, it is not desirable to implement special methods for communication with each system in the upper layers of DSAM. Instead, a communication layer is worthwhile that abstracts from the different native communication protocols. This layer should provide an interface with which messages can be sent to individual SPS entities addressed by a unique ID that is independent of addresses used in specific networks. Automatic routing of messages is necessary, because a message may pass more than one network on its way from sender to receiver. E. g. a message from the DSAM controller to a WSN node must be first sent through the stationary network to the host, to which the WSN is connected. Afterwards it has to be transmitted to the gateway node of the WSN from which on it must be sent to its actual receiver. Eventually, this network layer must be suitable for implementation on all devices in the network of SPSs, i. e. on powerful workstations as well as on tiny, resource-constrained sensor nodes.

Although resources available on small embedded systems like sensor nodes increase, they are still very limited. E. g. in large-scale deployments, very low-cost sensor nodes are needed for economic reasons. There are also domains that impose strong limitations on sensor weight and size. This might be the case when sensor nodes are fixed to animals for biological research. Such restrictions limit the size of batteries while the nodes should be running for long time without human intervention. Thus usage of sensor nodes with only a few kilobytes of memory and limited computational power will remain necessary in future deployments. Implementing involved communication protocols for application in the WSN is therefore not appropriate.

A. Related Work

There are several publications that deal with communication in heterogeneous sensor networks. This section presents a brief summary of projects related to our setting.

[1]http://www.rdsp.uni-erlangen.de/

A development kit for building in-network query applications is provided by the Corona architecture [3]. Queries are entered into a central server. The server generates a set of tasks from each query and distributes them to nodes in the network. Integration of hosts in a stationary and wireless network is not examined. Furthermore Corona architecture is based on and limited to powerful SunSPOT sensor nodes whereas we are also investigating more resource constrained ones.

The *Global Sensor Network* (GSN) project [4] offers a middleware that integrates heterogeneous WSNs at network level. Each sensor network is abstracted by a virtual sensor that produces one data stream. Hereby GSN gets a homogeneous view on possibly heterogeneous sensors. In contrast to the article at hand, seamless data transfer between stationary and wireless networks is not considered. Moreover addressing an individual WSN node is not designated.

REED [2] integrates the SPE Borealis [5] for stationary networks with the WSN TinyDB [6]. The integration is specialized for those two systems and cannot be easily adopted to integrate additional SPSs. Integration of REED and Borealis is only covered superficially as the article focuses on the development of an optimized algorithm for joins in sensor networks.

The *Delay Tolerant Networking Architecture* (DTN) [7] allows for integration of different network types like TCP/IP networks with native sensor networks. A special communication protocol is used and network borders are not visible to the communicating entities. DTN supports general communication and therefore must be supported by the operating system's network stack whereas in our setting a concept tailored to communication in the context of SPS integration is needed that runs on top of the network mechanisms present on a target platform.

B. Contribution

We present an overlay network for seamless communication across networks of different type and capabilities. Because it can be tailored individually to the different hosts and communication paths, the integration of highly diverse devices becomes possible while using the most efficient communication method available between two certain hosts. Moreover the concept is suitable for workstations as well as for memory- and performance-limited sensor nodes. We also developed an architecture and implemented the overlay network in form of a proof-of-concept-study for the integration of a WSN into DSAM.

The remainder of this article is structured as follows: The next section sketches DSAM. Its extension by the overlay network enables collaboration of SPSs situated in different networks. In Section III, we first discuss different approaches to communication over heterogeneous networks. Then we propose our cross-platform overlay network and describe its modular architecture. Furthermore our approach to global addressing and possibilities for tailoring the overlay network to resource-constrained platforms are detailed. In Section IV we

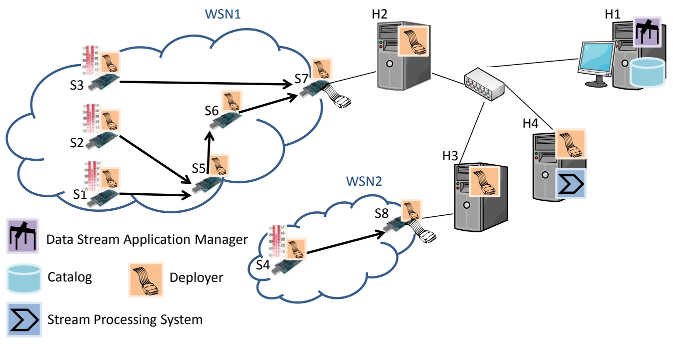


Fig. 1. System overview

present evaluation results. The article closes with a conclusion of our work.

II. DSAM Overview

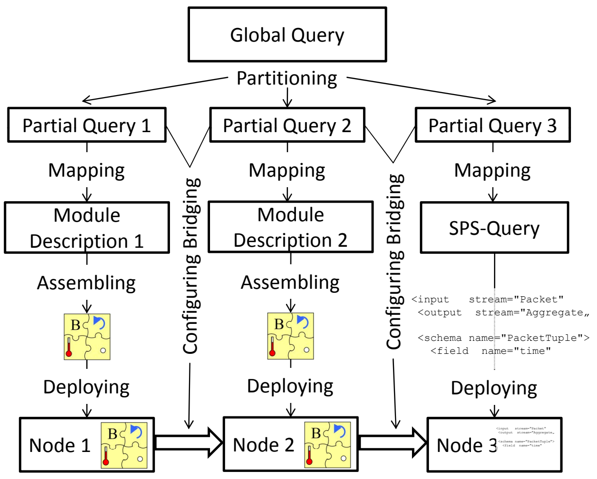
A top-down approach for the integration of sensor networks and the distribution of global queries requires a management component that accepts the definition of these global queries and propagates the appropriate partial queries for in-network stream processing. This section describes our system model of data streams, queries, and stream-processing components. It continues with the query management in DSAM, the central component that manages the deployment and maintenance of queries. DSAM does not do stream processing itself but integrates available SPS entities including both SPEs and WSN nodes to a federated stream-processing environment.

A. System Model of Data Streams, Queries, and SPSs including WSNs

Figure 1 shows a system overview of our approach. DSAM interacts with all participating SPEs and WSN nodes. The SPEs are installed on standard PCs interconnected via standard Internet protocols. Further, we integrate the WSN as a collection of individually configurable sensor nodes. A so-called deployer runs on all hosts that administrates the local SPSs according to commands received from the DSAM controller.

All SPEs and WSN nodes have potentially different capabilities. A common set of capabilities is grouped using the notion of a type. An SPS type defines capabilities such as query language and a core set of operators. A concrete SPE or WSN node may have additional individual capabilities such as special operators and characteristics (unique address, lifetime, etc.). All SPSs have unique addresses, and sensor nodes belong to a particular WSN. The benefit of the proposed overlay network is enabling the direct communication among all kinds of sensor nodes and SPEs.

Streaming data sources must have a unique address and a schema. Like in relational databases and most SPEs that refer to a relational model of data stream items, we use typed data-stream items that consist of a vector of attributes. DSAM supports the deployment of global abstract queries, i.e., an abstract query identifies input and output streams as well as the conceptual operators that manipulate the streams. The abstract



```
<input   stream="Packet"
 <output  stream="Aggregate„

 <schema name="PacketTuple">
   <field  name="time"
```

Fig. 2.   Mapping of Queries

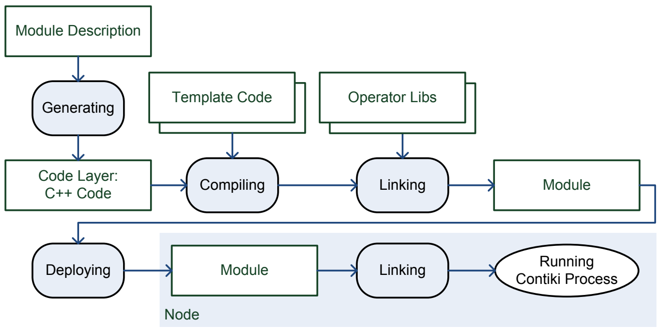


Fig. 3.   Module generation

query and the streams are the model of a data-stream query. Its concrete counterpart is a set of concrete queries representing partial queries that are distributed in an appropriate way. They use the query language or the programming model of the corresponding SPE or WSN node.

B. Query Management

Users interact with DSAM by writing global queries in the *Abstract Query Language* (AQL). AQL features a set of abstract operators with precisely defined semantics. Examples of AQL and its explanation can be found in [8], [1]. Figure 2 shows the overall deployment process of a global query on different kinds of nodes.

The operator-distribution process splits a global query into several so-called partial queries and decides which partial query is to be deployed on which node. Thus, the partial queries are the unit of distribution. A large number of constraints limits the number of possible distributions. Within these constraints, we optimize for performance by minimizing cost functions. The most important structural constraints are:

- Input streams come into existence at a certain place/node.
- Some nodes realize some abstract operators that others do not (e.g. because they do not have an operator implementation with compatible semantics).
- Nodes have individual capacity and performance behavior. The assigned tasks of a node must not exceed its capacity.
- Nodes' connections have reachability constraints and performance properties. Data may only be routed along existing connections having certain capacities.

In [1], we provide further details about the query partitioning.

After the query-partitioning step, DSAM maps partial queries to the corresponding platform-specific query languages. The results of the partitioning process are partial queries that can be deployed on the according node, i.e. the node must provide all necessary operators. Like the WSN platform Contiki considered in our experiments, some stream-processing components do not support query languages. For those nodes, binary modules are generated that represent those queries. DSAM generates source code from the model describing the partial query.

Figure 3 gives an overview of the whole process starting with the partial query as the module description and ending with the running process on the target node. In a first step, C++ source code is generated corresponding to the partial query. The query mapper writes glue code that takes care of calling the operators in the correct order and passing tuples from one operator to the other. Thus, the inner streams of the partial query graph represented by the module description are mapped to the generated code. It is compiled together with a code template for the query module and the code of compile-time optimized operators. The code template contains all the standard code that is independent of special queries. In the next step, the resulting object file is linked with libraries containing implementations of operators for static linkage and stubs for resident operators. The originating module is then transferred to the target node in the deployment process. On the node, it is linked with operating system libraries and is executed, which results in a running process ready to serve incoming tuples. In our earlier work [9], we demonstrate a linker for size optimization.

Technical integration is achieved during mapping and deployment with the help of the knowledge about platforms and connections (Figure 2). The addresses of inner streams among SPS entities can be directly derived from the knowledge about partitioning.

III. Communication Across Network Boundaries

For integration of WSNs with DSAM, a way to communicate with sensor nodes is needed. TCP/IP communication is used for message exchange with the deployer on hosts in stationary networks. The deployer listens on a predefined port for incoming requests. The DSAM controller connects to that port when communicating with the deployer. Messages are serialized to a format suitable for network transport, delivered to the recipient which deserializes the received data. Afterwards instructions or tuples contained in the message are processed and an acknowledgement message may be sent back. TCP/IP may not be available for communication with small sensor nodes by default. Therefore, passing messages to nodes in WSNs must be treated specially.

Because DSAM should be able to integrate arbitrary WSNs and possibly also SPEs in wired networks that run special protocols, it is not reasonable to adapt the whole system for each new sensor system to be used. As far as query distribution and optimization are concerned, different SPSs, regardless if small sensor nodes capable of query processing or full-fledged SPEs running on high-performance workstations, only differ in supported operators, number of operators that can be executed in parallel and costs for query execution and data transmission. For integration of arbitrary SPSs with little effort, a layer is needed that abstracts from the native methods to communicate with each type of SPS. With such an abstraction, only a deployer for the target system has to be implemented and the communication layer to be extended for support of an additional system. Otherwise development and especially maintenance costs would highly increase with each new supported system.

We have identified the following requirements for a communication layer for integration of arbitrary SPSs:

1) Possibility for message exchange between two hosts potentially situated in networks of different type
2) Identical message format for all recipients at upper application layers
3) Global addresses for all SPSs irrespective the type of network they are connected to
4) Automatic routing of messages across network borders based on metadata about the global network topology
5) Easy expandability for support of additional network types
6) Possibility for resource saving implementation

The first requirement is apparent, as communication between two arbitrary SPSs should be enabled. As there should not be any special handling necessary at layers above the communication layer to send instructions or tuples to arbitrary SPSs, messages must have a common format for all recipients. This implies requirement 2). Requirements 3) and 4) derive from the request that no knowledge, about the network the receiver of a message belongs to, should be necessary. Therefore it must be possible to globally address the receiver, and messages must be forwarded to other networks if necessary. The cause for requirement 5) is described above, as DSAM should be easily extensible. For integration of resource constrained sensor nodes, it must be possible to implement the necessary services of the network layer having few computing effort and small memory footprint. Omitting some of the services usually provided by the network layer on small platforms may be tolerated. E. g. the implementation of full routing capabilities is not necessary in a sensor network. It will usually not be used for transit.

A. Discussion of Different Approaches

This section presents a comparison between two approaches for integration of heterogeneous networks. Pros and Cons of the concepts are given as well as a comparison with the requirements stated above.

1) Implementation of TCP or UDP over IP for all Networks: A simple approach from the DSAM point of view is the implementation of the TCP or UDP and IP protocol for all networks. For networks that do not inherently support IP, an emulation has to be implemented based on the native protocols.

The main advantage of such a solution is that all hosts can be approached using the same communication mechanism. However there are some major disadvantages. As DSAM sends serialized messages, recipients must be able to deserialize the data received. As the same data format should be used for all hosts (requirement 2) the most restrained system determines the format. Even worse, different hosts might have conflicting requirements to message formats. The same issue exists with the decision between UDP and TCP as transport protocol. It must be made for the whole network. So it is not possible to use reliable TCP for the communication between hosts where the overhead of TCP is reasonable and to use UDP between hosts where unreliable communication may be tolerated for the sake of smaller resource consumption.

Resource consumption is important on small sensor nodes. Overhead in the packets caused by the IP protocol is addressed by techniques like 6LoWPAN [10]. But also RAM and ROM consumption of the communication stack must be considered. Even *micro IP* (uIP) [11] in its IPv4 version, an implementation that is specifically tailored for small embedded systems, has a memory footprint that is not negligible on platforms with high memory constraints. As stated in [12], usage of TCP/IP in WSNs may also be disadvantageous because it was originally designed for stationary networks.

2) Overlay to Abstract from Underlying Physical Network Topology and Protocols: Problems of the aforementioned approaches can be solved by introducing a thin overlay network layer that abstracts from the natively provided network protocols. Other than with the TCP/IP approach, the native protocols are directly used to send the data. Merely global addressing and gateways between the different network types are added. Routing information is extracted from the metadata that is anyway managed by the DSAM controller. It informs gateway hosts about necessary routing information via messages.

In Figure 1, two WSNs are connected to the stationary network via serial interfaces of their gateway nodes. Sensor nodes inside of a network can directly communicate with each other using a native multi-hop protocol. The stationary network is IP based. S7 and S8 are gateways between their WSN and the serial line connection. Accordingly H2 and H3 are gateways between the stationary network and the serial line. For a network spanning communication, e. g. between H1 and S3, the messages are routed by the overlay network components. In detail, a message from H1 is first transmitted to H2 which forwards it to S7 over the serial line. S7 finally forwards the message to S3 using the native protocol of the sensor nodes.

A major benefit of this concept is the exploration of native protocols. No additional complexity and costs are added to the communication between two hosts in the same network. Solely global addresses must be mapped to those used by the network, e. g. IP address and port, but determination of the recipient network address, is necessary without overlay network, too. Moreover, special properties of certain networks are handled by the native protocols. So, the overlay network does not need to deal with e. g. dynamics inside WSNs as long as interconnection points between networks do not change. Nevertheless all hosts are able to communicate directly with each other. No special treatment for non-TCP/IP networks is necessary in the upper application layers.

Using a well-thought-out architecture gives the possibility to extend the overlay network layer with support of additional network interfaces and protocols in an easy and maintainable way. An essential point is that gateways can provide more functionality than merely forwarding data. Changes to the payload are also possible. This allows e. g. for the use of a certain serialization mechanism in the stationary network and one that is tailored to the needs of resource constrained systems in a WSN. Conversion is performed by the gateway hosts. Likewise messages or parts of messages that are not needed in a certain network can be discarded. E. g. information needed for transit routing does not need to be forwarded to a network that is not intended to be used for transit. Omitting unnecessary data saves resources and energy. As we will show in the evaluation (Section IV), integration of an overlay network layer into DSAM deployers has a much lower memory footprint than uIP.

The raise of complexity that may be caused by integration of an overlay network into the DSAM system could be mentioned as counter-argument. But with a thoughtful design and application of state-of-the-art software engineering techniques this fact can be well handled.

Considering the arguments presented so far, we decided for the overlay network to be the best solution for the given requirements.

B. Overlay Network

In this section we detail our approach for integration of heterogeneous networks using an overlay network.

Definition. In this article we use the term overlay network in the following sense: The overlay network provides the ability to hosts to communicate with other hosts that are situated in the same network or in another network that is directly or indirectly connected to a host in their network. This ability is achieved by a globally unique address for each participating host and forwarding and translation of messages from one network to another by hosts that are connected to two different networks.

The overlay network layer provides the ability for communication between two deployer instances running on arbitrary hosts. Therefore global addressing must be provided. The actual target of a control command or a data stream is not the deployer itself but one of the SPSs running on the deployer's host. Commands for installing or starting queries should be performed on a certain SPS and tuple streams are to be processed by a certain SPS. The overlay network supports this by using global unique addresses for each SPS. Since all SPS entities must be known and distinguished for query distribution and optimization, they are listed in the metadata repository with unique IDs anyway. These IDs can be used for communication purposes, too.

Communication services are provided to upper layers by means of an interface. This includes methods for sending messages to an SPS ID and for listening for incoming messages addressed to the ID of an SPS on the local host.

For sending a message to a certain host, upper layers call a method of the interface with the ID of the addressed SPS. The overlay network searches the local routing table for an entry with the target ID. This entry holds the interface to be used and the parameters necessary to establish a native connection either with the recipient itself or with an intermediate node that has to forward it to another network. For a recipient directly available through TCP/IP parameters would be the IP address and port number of the recipient. For a message to a sensor node, e. g. S1 in the scenario of Figure 1, from a host in the stationary TCP/IP network, the parameters would be IP address and port of the host to which the WSN is connected (H2). The local routing table is filled by messages from DSAM controller. More details are given in Section III-C.

At startup of a deployer instance or the DSAM controller the overlay network layer starts listening for incoming connections on all interfaces that are supported on the host. For TCP/IP, this means opening a predefined port; for a serial line connection, a read operation is started. When a new connection attempt is received, the type of connection is determined by reading the first message arriving. Connections may be dedicated to one of the SPSs on the local host. Those connections are forwarded to the upper layers if they are listening for incoming connections. Otherwise the connection is aborted. Connections might also transport information for the overlay network layer like routing information. The handling of routing information is explained in Section III-C. Incoming connections may also be addressed to an SPS on a foreign host and have to be routed by the local host. In this case a connection addressed to the target host of the connection is established in the same way as when sending messages from the local host. The new connection might end at the actual target of the forwarded connection or at another intermediate routing point depending on information in the local routing table. A thread is started that receives all messages arriving at one connection, accomplishes necessary conversions, and sends the message on the other connection. When one of the two connections closes, the other one is closed too. Closing a connection by either of the two communication parties propagates through all intermediate connections.

To accomplish the goal of easy expandability, support for a certain network is encapsulated in a plugin in form of a so-called adapter. At system startup of the DSAM controller or a deployer all available adapters are loaded and set to listen for incoming connection attempts. When a connection should be established the corresponding adapter is determined by the local routing table and asked to create the new connection. Incoming messages are deserialized and put into a message object in generic format. This object is passed to upper layers or, if the connection is just routed, passed to the adapter for the outgoing interface.

Additional interfaces and networks can be supported by implementing an adapter that deals with the low-level details of sending and receiving in its network. Furthermore serialization and deserialization as well as any transformations of message content are provided by the adapters. Through utilization of a common network-independent format for message exchange between components like adapters and higher-level ones, support for new networks can be added without changes to existing parts of the system.

C. Determination and Dissemination of Routing Information

Topology of the whole network, including both stationary and wireless hosts, has to be known centrally in this approach. This data is needed for query distribution and optimization. Routing information can also be obtained from this metadata. Each deployer instance on a host has a local routing table to hold routing information that is locally needed. An item in the routing table contains the target of a connection, the adapter that has to be used for communication with the next routing point towards the target or with the target itself and parameters necessary for initiating a connection with this host. For a serial line adapter, parameters may be e. g. the device name of the serial interface and the baud rate. Local routing tables are empty at startup and the IDs of local SPSs are unknown to the deployer.

The DSAM controller is responsible for disseminating necessary routing information before it sends a command to a deployer that has not received commands before and before the forwarding of stream data between two SPSs is initiated. Therefore, the shortest path between sender and receiver is determined. A message containing the routing information is composed and sent to the sender. The message states the information for the routing tables for all hops that are involved in the communication between sender and receiver. The sender adds the first entry of the messages to its routing table and deletes it. Afterwards the message is sent to the receiver using the routing table entry currently created. The next hop does the same as the sender and forwards the message. As last entry the routing message contains the ID of the receiving SPS. When the message arrives at the receiver only this entry is left telling it that the SPS with the stated ID is running locally. By this way only one message is needed to setup a complete routing path over several hops. Figure 4 shows this process for setting up a route between H1 and S2 in the scenario given in Figure 1. S2 has the SPS-ID 5.

D. Tailoring to Resource-Constrained Platforms

On resource-constrained platforms like wireless sensor nodes it is essential to keep memory and CPU usage low. For this some functionality may be omitted. Other work can

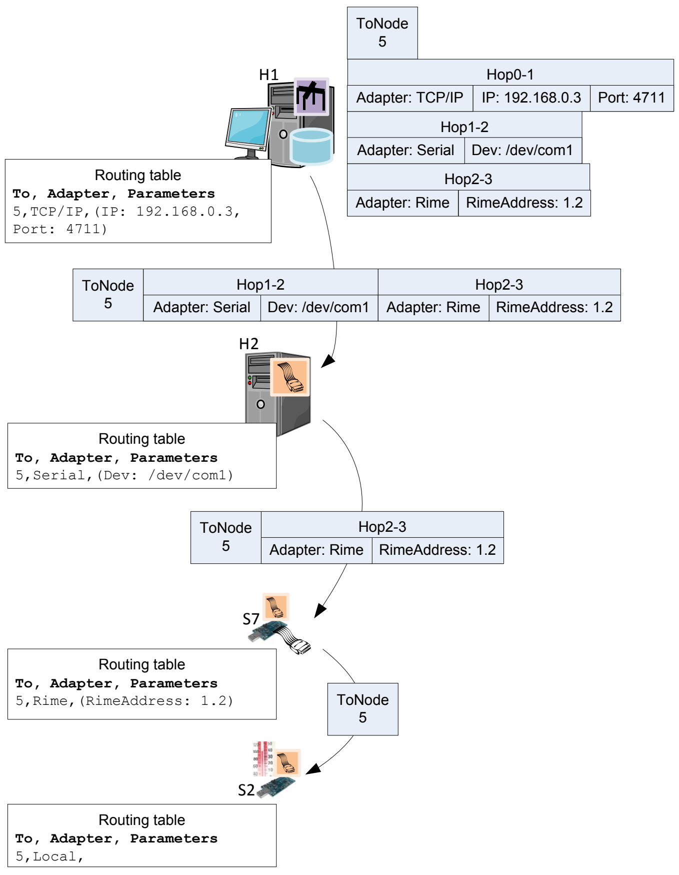


Fig. 4. Schema of routing information dissemination

be prepared and supported by the adapter on the stationary gateway node to relieve the sensor nodes. Since messages are routed and not just bridged, adapters have all possibilities in message processing. This section presents some approaches.

Routing tables may allocate lots of memory when the node communicates with many different recipients. So moving routing tables out of the wireless network is advantageous. WSNs are usually not used for transit and native communication protocols often provide multi-hop communication. Connections that would be normally established by the deployers on the sensor nodes can be established by the adapter on the stationary gateway instead. In this case, the global IDs used in the overlay network can be translated to native addresses of the sensor network by the adapter on the stationary gateway host. Messages are transmitted to the wireless gateway node that can directly forward them to the recipient. No lookup of the recipient's native address is needed by the overlay network layer on the wireless node. So keeping a routing table can be omitted. This saves both memory consumption and CPU power. Additionally, messages with routing information need not to be forwarded to the WSN. E. g. transmissions between H2 and S7 and between S7 and S2 could be omitted in the routing information dissemination scenario of Figure 4. Less messages implies less energy consumption and a longer lifetime. Messages from sensor nodes to hosts in the stationary

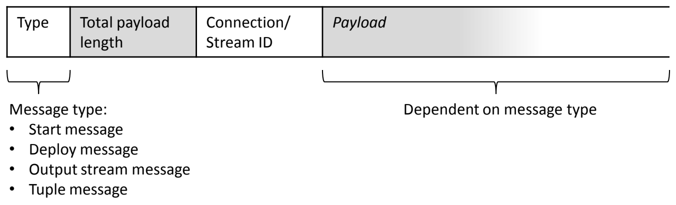


Fig. 5. Message format in the Contiki TelosB WSN

network are generally sent to the gateway node, which sends it to the stationary network. On the stationary gateway, messages are reassigned to connections and forwarded.

Since adapters are responsible for serializing messages, a format can be chosen that is most suitable for processing on the sensor nodes. Furthermore unnecessary parts of messages may be omitted or additional parts added. Certain messages can even be handled by the adapter itself instead of forwarding them to the actual recipient. E. g. a field with a connection ID may be added to preserve the mapping between messages and connections on connectionless protocols.

Figure 5 shows the message format used inside the WSN in our prototype implementation for TelosB nodes running Contiki OS. The type field states the type of payload contained in the message. SPS-IDs are omitted since messages are directly send to the receiving node via Contiki's native network stack. Only messages send to the gateway node from the stationary host via serial line need to state the receivers address. Once inside the WSN this information is passed to the native communication layer and is no longer needed inside the application message data. Because the native communication protocol chosen does not provide connections, a connection or stream ID is included for control or data stream connections respectively. Furthermore the first packet of a message states the total payload size. This enables the receiver to determine if the message was split into several packets due to packet size constraints.

IV. Evaluation

We implemented the overlay network demonstrating integration of WSNs in a federated stream-processing environment. For the proof-of-concept study, we chose a WSN of TelosB sensor nodes running the Contiki operating system [13]. TelosB nodes are based on an MSP430 microcontroller and provide 48 KiB of flash ROM and 10 KiB of RAM. Contiki provides several native communication protocols and uIP. In a first step, we integrated an overlay network layer as described in the previous sections into DSAM, replacing the existing solution for TCP/IP communication between deployers. The necessary information about network topology is taken from the DSAM metadata catalog for this prototype implementation. The only network adapter implemented so far was one for TCP/IP communication. By deploying several queries, we ensured that the overlay network adequately replaces the former communication solution

As next step, we added a network adapter for communication with the WSN gateway node over a serial line and

```
struct OverlayNetwork {
  // Accept incoming connections
  static Connection accept();

  // Establish new connection
  static Connection getConnection(unsigned int spsId
      );
  // ...
};

struct Connection {
  virtual void sendMessage(const Message& cmd);
  virtual void readMessage(Message*& cmd);
  // ...
};
```

Listing 1. Extract of the interface provided by the overlay network to higher application layers

developed a deployer for Contiki together with an adapter for communication inside the WSN using a native Contiki multi-hop protocol. We implemented the techniques presented in section III-D. Because of the overlay network layer abstracting from the native communication protocols, no existing code had to be changed. The serial line adapter was placed as an additional dynamic link library into the search path of the overlay network layer. Adapters to integrate devices that use other connections and protocols, e. g. Bluetooth, may be added in the same simple way. Listing 1 shows an extract of the interface provided by the overlay network to higher application layers. They only have to deal with message objects, containing payload in a network independent format, and the globally unique SPS entity ID. Higher application layers do not need any knowledge about the receiver's platform or network connection. So the aim of global addressability, easy expandability, and possibility of resource conserving implementation were achieved.

We evaluated network spanning communication in the scenario shown in Figure 1. We simulated the WSNs with Cooja instead of using real hardware sensor nodes. Cooja is a simulator for WSNs with sensor nodes running the Contiki operating system. It executes unmodified versions of the applications built for real hardware nodes. The serial link connection is substituted by pipes. The advantage of the simulator is the possibility to obtain additional information necessary for detailed testing that is not easily available on hardware sensor nodes. In this set-up we successfully verified the exchange of messages between the DSAM controller and deployers on both sensor nodes and stationary hosts. To evaluate transit routing, the stationary network was split into two parts with one host being connected to both networks. We successfully tested communication between a host in the first stationary network and a sensor node in a WSN connected to the second stationary network. This shows that the requirements of message exchange between hosts on arbitrary networks and automatic routing are fulfilled.

A code review indicates that the overlay network imposes only little additional costs for format conversion compared to

TABLE I
COMPARISON OF MEMORY FOOTPRINT OF PURE CONTIKI, CONTIKI WITH DEPLOYER AND OVERLAY NETWORK, AND CONTIKI WITH DEPLOYER AND UIP

	Memory footprint		Available memory (Improvement to uIP)	
	ROM (text)	RAM (data + bss)	ROM	RAM
Totally available	48 KiB	10 KiB		
Pure Contiki	22.3 KiB	5.2 KiB	25.7 KiB (656 %)	4.8 KiB (45 %)
Overlay network	38.2 KiB	5.5 KiB	9.8 KiB (188 %)	4.5 KiB (36 %)
uIP	44.6 KiB	6.7 KiB	3.4 KiB (0 %)	3.3 KiB (0 %)

uIP. Messages have to be serialized at the sender anyway. The same holds for deserialization at the receiver. Overhead could only be introduced at gateways by conversion of the serialization format. But format conversion before and after serial line transmission is also necessary in uIP due to limitations of the Contiki serial interface. Moreover the format conversion gives the possibility to use the best suited format in each network, which is worth the small conversion costs.

To show the advantage in memory consumption of the utilization of native specialized protocols with the overlay network instead of TCP/IP in sensor networks a second version of the deployer for sensor nodes was developed that uses uIP for communication. uIP uses the *Serial Line IP* (SLIP) protocol on serial links. Nodes have unique IP addresses and can be directly addressed from the stationary network.

Table I shows the size of text and data segments with both approaches. The results for pure Contiki represent the memory footprint of the operating system itself without added functionality. The following two rows show the memory consumption by Contiki with deployer and overlay network or the IPv4 version of uIP respectively. The deployer's size is mainly determined by the size of the dynamic linker which is always needed independent of the communication layer. uIP increases the size of the text segment leaving merely 3.4 KiB for operators and sensor reading. Also the data + bss segment consumes additional 1.2 KiB of RAM. Considering the small amount of memory available, the memory saved by the overlay network approach is essential for the ability to accomplish elaborate data-stream processing on such small sensor nodes.

V. CONCLUSION

In the context of a federated data-stream environment project, it has been possible to provide a carefully designed overlay network for the sensor nodes and their connections to SPEs for a simple addressing mechanism and a very small footprint on the sensor nodes. We compared alternatives for interconnection of different networks and argued why abstraction is fitting best by evaluating our resource efficient approach compared to uIP. Requirements especially for integration of small, resource constrained sensor nodes were identified. Our solution provides a modular architecture that enables tailoring in order to reduce ROM and RAM memory footprint. In the evaluation, we offer concrete measurements for the resource consumption of different approaches.

Up to now, development of in-network streaming applications had to be deferred as the available memory (uIP in Tab. I) did not meet the requirements. Streaming applications using our overlay network could be evaluated successfully using a strongly reduced set of operators. Future work will consist of the development of additional operators for stream processing on sensor nodes, the analysis of metadata requirements for automatic management of WSNs, and the development of network adapters for additional communication protocols.

REFERENCES

[1] M. Daum, F. Lauterwald, M. Fischer, M. Kiefer, and K. Meyer-Wegener, *Wireless Sensor Networks Technologies for the Information Explosion Era*, ser. Studies in Computational Intelligence. Berlin Heidelberg, Germany: Springer, 2010, no. 278, ch. Integration of Heterogeneous Sensor Nodes by Data Stream Management, pp. 139–172.

[2] D. J. Abadi, S. Madden, and W. Lindner, "REED: Robust, Efficient Filtering and Event Detection in Sensor Networks," in *31st Conference on Very Large Data Bases (VLDB)*, Trondheim, Norway, Aug. 2005.

[3] R. Khoury, T. Dawborn, B. Gafurov, G. Pink, E. Tse, Q. Tse, K. Almi' Ani, M. Gaber, U. Röhm, and B. Scholz, "Corona: Energy-Efficient Multi-query Processing in Wireless Sensor Networks," in *Database Systems for Advanced Applications (DASFAA)*. Tsukuba, Japan: Springer, Apr. 2010.

[4] K. Aberer, M. Hauswirth, and A. Salehi, "Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks," in *International Conference on Mobile Data Management (MDM)*, Mannheim, Germany, May 2007.

[5] D. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.-H. Hwang, W. Lindner, A. S. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik, "The Design of the Borealis Stream Processing Engine," in *2nd Biennial Conference on Innovative Data Systems Research (CIDR)*, Asilomar, CA, USA, Jan. 2005.

[6] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: An Acquisitional Query Processing System for Sensor Networks," *ACM Transactions on Database Systems*, vol. 30, pp. 122–173, 2005.

[7] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*. Karlsruhe, Germany: ACM, Aug. 2003, pp. 27–34.

[8] F. Dressler, R. Kapitza, M. Daum, M. Strübe, W. Schröder-Preikschat, R. German, and K. Meyer-Wegener, "Query Processing and System-Level Support for Runtime-Adaptive Sensor Networks," in *16. GI/ITG Fachtagung Kommunikation in Verteilten Systemen (KiVS)*. Kassel, Germany: Springer, Mar. 2009, pp. 55–66.

[9] M. Strübe, R. Kapitza, K. Stengel, M. Daum, and F. Dressler, "Stateful Mobile Modules for Sensor Networks," in *6th IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS)*, vol. LNCS 6131. Santa Barbara, CA, USA: Springer, Jun. 2010, pp. 63–76.

[10] G. Mulligan, "The 6LoWPAN Architecture," in *Proceedings of the 4th Workshop on Embedded Networked Sensors (EmNets)*. Cork, Ireland: ACM, Jun. 2007, pp. 78–82.

[11] A. Dunkels, "Full TCP/IP for 8-Bit Architectures," in *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys)*. San Francisco, CA, USA: ACM, May 2003, pp. 85–98.

[12] A. Dunkels, T. Voigt, J. Alonso, H. Ritter, and J. Schiller, "Connecting Wireless Sensornets with TCP/IP Networks," in *Proceedings of the Second International Conference on Wired/Wireless Internet Communications (WWIC)*, Frankfurt (Oder), Germany, Feb. 2004.

[13] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki – a Lightweight and Flexible Operating System for Tiny Networked Sensors," in *29th Annual IEEE International Conference on Local Computer Networks (LCN)*, Tampa, FL, USA, Nov. 2004, pp. 455–462.