Filter-enabled Binary XML Dissemination in Embedded Networks

Sebastian Käbisch[1,2] and Richard Kuntschke[1]
[1]*Siemens AG, Corporate Technology, 81730 Munich, Germany*
[2]*University of Passau, Chair of Distributed Information Systems, 94032 Passau, Germany*

Keywords: Binary XML, EXI, Embedded Networks, XPath, Filtering

Abstract: Efficient data dissemination in distributed systems is a challenge that can be tackled by sharing common data and processing results among multiple queries. Doing so in an effective manner helps to save network bandwidth and computational resources. This is especially important in embedded networks where such resources are often extremely scarce. Disseminating resource-intensive XML data in embedded networks has been enabled by using binary XML technologies such as W3C's EXI format. In this paper, we show how filter-enabled binary XML dissemination in embedded networks helps to further reduce resource demands. Thus, through the suitable placement of pre- and post-filters on binary XML data, bandwidth on network connections and computational resources on nodes can be saved. Consequently, more data can be processed with a certain amount of available resources within an embedded network.

1 INTRODUCTION

Efficient data dissemination is an important task in distributed systems where data needs to be transferred from data sources to data sinks located at different places within the system. Sharing common data and processing results among multiple data sinks helps to save network bandwidth and computational resources. While such approaches reduce resource usage in any distributed system, their usage is crucial in embedded networks (fundamental in, e.g., building or industrial automation, automotive industry, and smart grid) due to the strictly limited resources such as from microcontrollers. Using binary XML techniques such as W3C's Efficient XML Interchange (EXI) format (Schneider and Kamiya, 2011) enables the dissemination of otherwise resource-intensive XML data in embedded networks. Additionally, using filter-enabled binary XML dissemination helps to further reduce resource demands.

A filter-enabled subscription mechanism in embedded networks reduces network traffic and unnecessary message processing at the client nodes. It optimizes data interaction between the service provider (data source) and the service requester/client (data sink) in terms of data novelty and supports an efficient execution of applications in embedded networks at runtime. Consequently, more data can be processed with a certain amount of available resources within an embedded network.

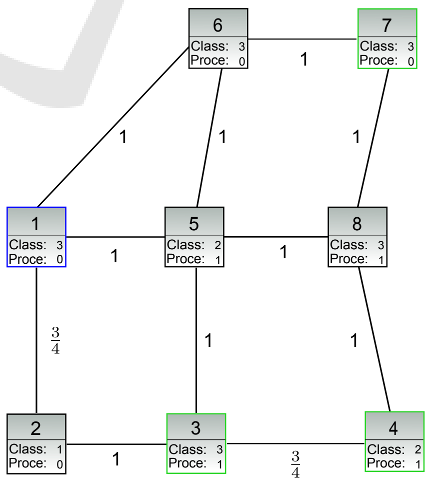


Figure 1: Example embedded network with a service provider (node 1) and three clients (node 3, 4, and 7)

An immediate evaluation at the node of service data origin prevents the dissemination of data that is outside of the scope of the corresponding service requesters. In the context of constrained embedded networks, however, a desired early evaluation at the node of service data origin sometimes cannot be realized. Two aspects support this observation: First of all, the available resources, especially when it comes to memory, are often not sufficient for installing an additional filter application. Secondly, vendors of em-

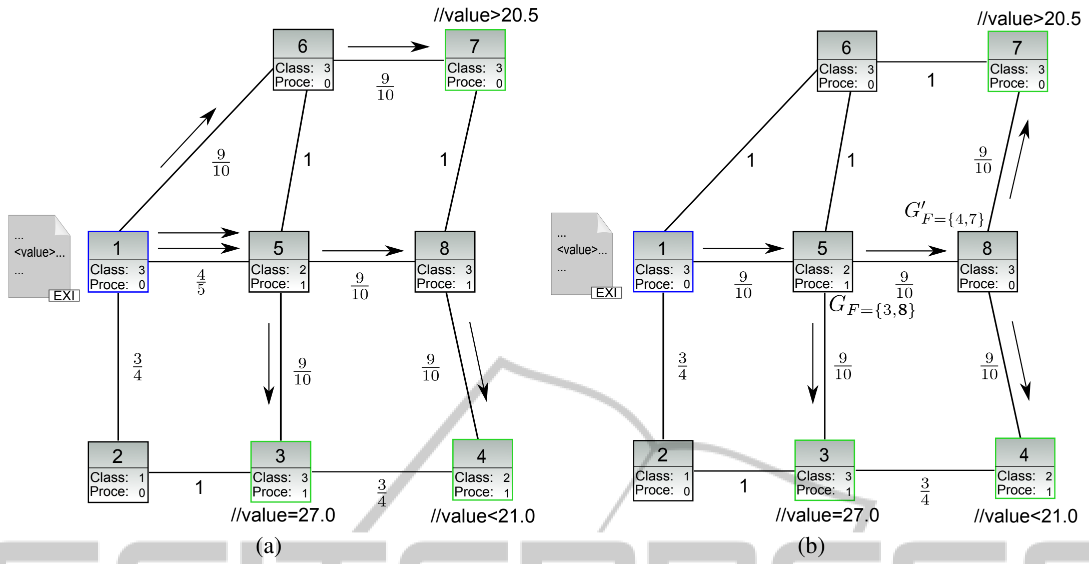


Figure 2: (a) Naive approach for data dissemination (b) Desired pre-/post-filter approach for data dissemination purposes.

bedded nodes do not offer the possibility of installing such filter applications.

In this paper, we introduce an approach for organizing filter-based service data dissemination in constrained embedded networks that takes into account resources such as device classes, device processing performance, and connection quality between nodes. The goal is to share relevant service data using binary XML whenever possible by using filters and subfilters, respectively. This reduces both network traffic and computational load on nodes. Basically, this approach leads to content-based routing at the application level based on binary XML content.

1.1 Problem Statement

Consider Figure 1 as an example embedded network that shows a simple distributed system with eight nodes and corresponding network connections. Nodes are categorized here into three classes (1 to 3) indicating their capabilities. The lower the class number, the more powerful the device. Class 1, e.g., corresponds to a consumer PC, class 2 corresponds to a hardware system with a resource complexity found in home network routers, and class 3 corresponds to constrained embedded hardware such as a microcontroller (e.g., ARM Cortex-M3[1] with 24MHz, 256kB RAM, and 16kB ROM). The boolean *proce* value indicates whether there are computational resources available at the corresponding node. A value of 0 indicates that there are no resources left for installing a potential application, while a value of 1 indicates available resources. Resources in this context may comprise memory as well as processing power. The numbers at the network connections indicate the available connection quality. Values closer to 1 indicate superior connection quality. In contrast, values closer to 0 indicate poor connection quality. Connection quality may be impacted by the technology of the underlying physical communication link, e.g., low-power wireless communication (e.g., IEEE 802.15.4 (IEEE, 2011)) vs. Ethernet, as well as by the current system state as indicated, e.g., by the current package loss ratio, delay, and available bandwidth of the communication link. Increasing network traffic on a communication link lowers the connection quality since, e.g., increasing traffic reduces the available bandwidth.

Now, let us assume a simple data dissemination scenario. In Figure 2(a), node 1 is the data source, sending XML-based data including a *value* information in EXI format, and nodes 3, 4, and 7 are the data sinks that are interested in the *value* information. The data sink at node 3 is only interested in data elements containing the value 27.0, node 4 is interested in data elements containing values less than 21.0, and node 7 is interested in data elements containing values greater than 20.5. Since node 1 does not have the opportunity to set up a filter mechanism (*proce* is equal to 0) to test client's relevance of a new data message, this node always sends an individual copy of the entire data message to each data sink and the nodes constituting the data sinks process the data accordingly (see Figure 2(a)). However, this leads to relatively high bandwidth usage on the affected network connections and requires considerable computational re-

[1]http://www.arm.com/products/processors/cortex-m/cortex-m3.php

sources at the data sink nodes, even if this message is not relevant anyway. To illustrate the impact of this naive dissemination approach in terms of connection quality, in Figure 2(a) in each involved dissemination link the connection quality value is decremented by the value $\frac{1}{10}$.

Now consider Figure 2(b), showing our pre-/post-filter approach for the same distributed network. All produced data is sent out by the data source only once. A pre-filter (G_F) at node 5 filters the data of relevance for nodes 3 and 8. A message for node 3 is only forwarded when the *value* in the data message is equal to 27.0. Node 8 receives only a message from node 5 when a *value* information is present in the message. A post-filter (G'_F) at node 8 is applied to evaluate the relevance for nodes 4 and 7. Consequently, node 8 forwards only the messages to node 4 when the *value* is smaller than 21.0. Node 7 only receives a message when the *value* is greater than 20.5.

Thus, compared to the naive approach, the requested data is transmitted to each data sink using less network bandwidth and less processing power overall in the system, since processing results can be shared among multiple data sinks. Also, we are able to distribute the computational load across capable nodes within the distributed system. We accept this approach even if the resources of a small number of node devices may be claimed by the filter placement. This is seen by the processablity of node 8 that is set to 0.

1.2 Paper's Contributions and Outline

This paper presents the following contributions:

- We shortly introduce a previously developed filter mechanism based on W3C's EXI format (Section 2). This mechanism constitutes the basis for our further work on filter-enabled binary XML dissemination in embedded networks.
- In Section 3, we present our new approach for efficient filter-enabled binary XML data dissemination in embedded networks. The approach helps to reduce bandwidth usage on network connections and computational load on nodes, thus allowing larger amounts of data to be transferred and processed with a certain amount of available resources.
- Finally, we present evaluation results showing the effectiveness of our approach (Section 4).

2 EXI FILTER MECHANISM

In this section we give a short introduction about the EXI format and the basic idea, how an EXI grammar can be transformed to a filter grammar for evaluating XPath expressions.

2.1 The W3C EXI Format

W3C, the inventor and standardizer of XML, faced the drawbacks of plain-text XML and created a working group called XML Binary Characterization (XBC) (Goldman and Lenkov, 2005) to analyze the condition of a binary XML format that should also harmonize with the standardized plain-text XML format as well as with the XML Infoset. The outcome was the start of the W3C Efficient XML Interchange (EXI) format, which gained recommendation status at the beginning of 2011 (Schneider and Kamiya, 2011). Mainly, EXI is a grammar driven approach that is applied to bring XML-based data into a binary form and vice versa. Such a grammar is constructed based on a given XML Schema where each defined complex type is represented as a deterministic finite automaton (DFA). Figure 3(a) shows an excerpt of a sample EXI grammar (set of automaton) G that can be used for encoding and decoding. This grammar reflects an XML Schema including the SOAP framework (Gudgin et al., 2003) with a status information within the Header part and request/response patterns of temperature and humidity information within the Body part. Please note that the *Root* grammar is a predefined grammar that occurs in each EXI grammar representation of arbitrary XML Schemas. It contains all entry points of all root elements in a given schema. Here, we only highlight in our context the relevant root Envelope of the SOAP message framework. In general, each DFA contains one start state and one end state, which reflect the beginning and the end, respectively, of a complex type declaration. Transitions to the next state represent the sequential order of element and/or attribute declarations within a complex type. Optional definitions (e.g., *choice*, *minOccurs* $= 0$, etc.) are reflected by multiple transitions and assigned an event code (EV). E.g., the SOAP Envelope message framework embeds an optional Header and a mandatory Body element (Gudgin et al., 2003). The equivalent EXI grammar representation, as can be seen in Figure 3(a) (Envelope Grammar), provides two transitions from the start state: one to the Header state and one to the Body state. For signalization, a one bit event code is used and assigned to the transition (EV(1) for the Header; EV(0) for the Body). Generally, the number of bits used for m transitions is deter-

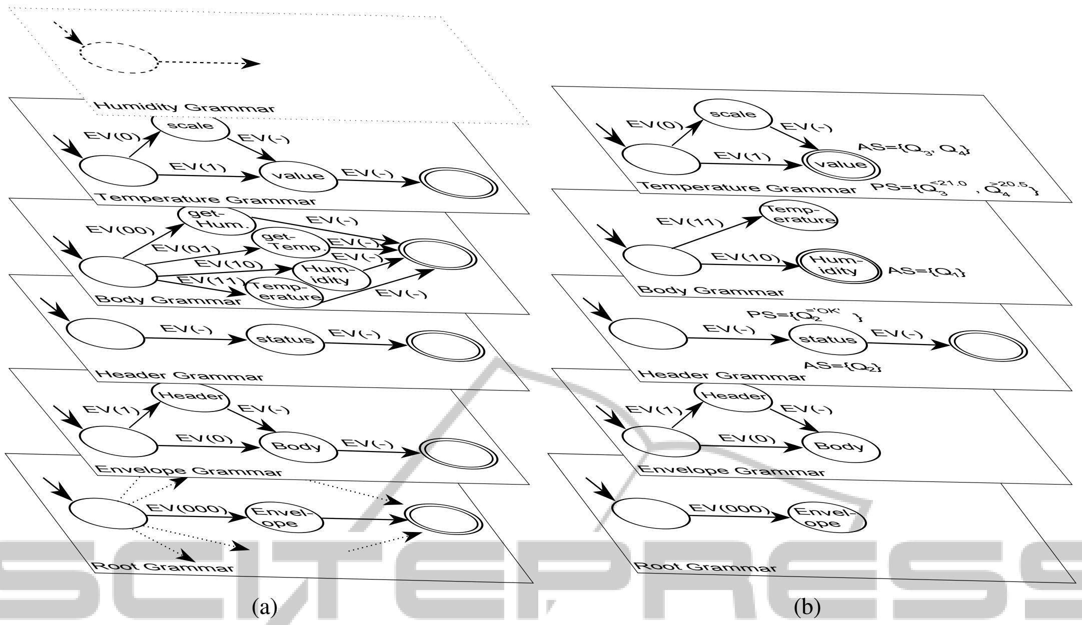


Figure 3: (a) EXI grammar G (excerpt) for encoding and decoding purposes (b) Filter Grammar G_F with $G_F \subseteq G$ based on the queries Q_1, Q_2, Q_3, and Q_4 for evaluation purposes.

mined by $\lceil log_2 m \rceil$. EV(-) on transitions indicates, no event code is required.
An example XML message snipped such as

<Envelope><Header><status>OK< /status>...

would be transformed to a

000 1 'OK' ...

EXI (bit-)stream based on the EXI grammar shown in Figure 3(a). This already shows, how compact EXI can be. In addition, EXI is a type aware encoder that provides efficient coding mechanisms for the most common data types (int, float, enumerations, etc.). There are use cases in which the EXI representation is said to be over 100 times smaller than XML (Bournez, 2009). Based on the high compression ratio and the opportunity to obtain the data content directly from the EXI stream, XML-based messaging is also feasible in the embedded domain, even if constrained devices are used (Käbisch et al., 2011).

2.2 EXI Filtering

EXI grammars build the bases for writing and reading binary XML data. Previous work (Käbisch et al., 2012) shows the functionality to create an efficient filter mechanism for binary XML data based on a number of service requesters by providing XPath expressions that address the desired service data occurrences and/or data value conditions. Two approaches which are feasible to constrained devices such as microcontrollers were presented: *BasicEXIFiltering* and *OptimizedEXIFiltering*. The *BasicEXIFiltering* operates on top of an EXI grammar and evaluates normalized XPath queries by means of binary XML. *OptimizedEXIFiltering* presents a more sophisticated approach; it maps all XPath expressions within an EXI grammar and removes all states and transitions which are not required for message evaluation. The outcome is a filter grammar denoted as G_F. Figure 3(b) shows such a constructed filter grammar G_F based on the queries

$$
\begin{aligned}
Q_1 &= //Humidity \\
Q_2 &= //status[text() =' OK'] \\
Q_3 &= //Temperature/value[text() < 21.0] \\
Q_4 &= //Temperature/value[text() > 20.5]
\end{aligned}
$$

applied on the data model represented by the EXI grammar G in Figure 3(a).
Applying the previously created EXI stream

000 1 'OK' ...

to the filter grammar G_F we would, at least, successfully find a match for the XPath expression Q_2. This is due to the remaining states and transitions in G_F that lead via the Envelope state (reading the bits 000 from the stream) and the Header state (reading the bit 1 from the stream) to the Header status state which is dedicated as a predicate state (PS) and as an accepting state (AS). A predicate state results to a predicate evaluation. Here, we have to evaluate whether the message contains the status value 'OK' which is also the case in our example. An accepting state represents a query match of a particular query (here Q_2). This is only true, if all aforegoing relevant predicate states resulted in a positive predicate evaluation.

The next section explains how such filter grammars can be used to realize an efficient binary XML data dissemination mechanism in constrained embedded networks.

3 DISSEMINATION ALGORITHM

This section presents our filter-enabled binary XML dissemination algorithm. First of all, we introduce our cost model and formalize the optimization problem (Section 3.1). Next, the dedicated algorithm is presented (Section 3.2). Finally, an example is considered for better clarification of how our algorithm works (Section 3.3).

3.1 Cost Model

Before we formalize our cost model we will formally define an embedded network in our context as described in Section 1.1 by $N_{emb} = (V, E, c, w, p)$. V describes the set of vertexes/nodes. Set E describes the set of edges/connections between two nodes. Function c with $c : V \rightarrow \mathbb{N}_{>0}$ associates the device class of a device node. The weight function w with $w : E \rightarrow \mathbb{R}_{[0,1]}$ describes the connection quality between two device nodes and p with $p : V \rightarrow \{0,1\}$ the processing capability of a node.

A newly installed application, including a service provider node (v_s) and service subscriber clients ($C = \{v_{c_1}, ..., v_{c_n}\}$), in an embedded network N_{emb} would typically lead to additional network traffic and processing costs. To keep this overhead as small as possible, we filter the data of relevance and share this data as long as possible on a determined dissemination path that avoids constrained device class nodes and uses connections with relatively good quality. Consequently, we have two metrics which have to be considered: device class with the processability (c and p) and connection quality (w). Putting this together, we define our cost function f for a given $N^T_{emb} = (V^T, E^T, c, w, p)$ with $N^T_{emb} \subseteq N_{emb}$ that only contains the nodes and transitions (spans a tree) from v_s to all nodes in C that is used for the data dissemination:

$$\begin{aligned} f(N^T_{emb}) := \alpha \cdot \sum_{v_i \in V^T} (c(v_i) + 1 - p(v_i)) \\ + (1-\alpha) \cdot \sum_{(v_i, v_j) \in E^T} \frac{1}{w(v_i, v_j)} \quad . \end{aligned} \tag{1}$$

The first summand formalizes the device class with the processing capability of a device node. A *hop-noise* value 1 is added to enable an influential decision when we also have only one class occurrences ($c = 1$) and each has processing capability ($p = 1$) in the network. The second summand represents the reciprocal connection quality. The $\alpha \in [0,1]$ is a weight factor that enables us to set up a more dominant part in the cost function: the device class ($\alpha > 0.5$) or the connection quality ($\alpha < 0.5$).

Using f we are able to formalize our optimization problem to find a subgraph N^T_{emb} of N_{emb} for a filter-enabled service data dissemination:

$$Minimize \quad f_T \tag{2}$$

subject to

$$\sum_{v_i \in V^T} p^T(v_i) \geq 1 \quad .$$

The inequality constraint specifies the occurrence of at least one node processing capability within N^T_{emb} that can be used to set up a pre-filter.

Unfortunately, for any constellation in N_{emb} we are not able to find an optimized solution in polynomial time. In the next section we are going to present an heuristic approach based on greedy algorithms that approximates an optimized N^T_{emb} for a filter-enabled service data dissemination.

3.2 Algorithm

We are now going to describe our filter-enabled service data dissemination algorithm, the *FilterEnabledDissemination* algorithm (see Algorithm 1), for installing a new application with a service provider and a number of service requesters, which takes into account the current resources of the embedded network.

As input, the algorithm takes an embedded network N_{emb}, a dedicated service provider node v_s, a set of service requesters (the clients) C and their corresponding queries Q, and the underlying data model of the service provider described in an XML schema *XSD*. Its outcome is a subnetwork N^T_{emb} of N_{emb} that represents the dissemination tree/path from v_s to all clients in C and a set F that consists of the selected nodes with pre- and post-filter properties. Essentially, the processing steps of the algorithm can be divided into three parts:

1. After determining the filter grammar G_F (see Section 2.2) by the *FilterGrammar* in line 1, a suitable pre-filter node is searched. Doing this, the *ClosestPreFilterNode* algorithm (line 2) is

Algorithm 1: *FilterEnabledDissemination.*

```
Require: $N_{emb} = (V, E, c, w, p)$, a service provider $v_s$, set of service requesters $C = \{v_{c_1}, ..., v_{c_n}\}$, set of queries $Q$ related to client's conditions, data model represented as XML schema $XSD$.
Ensure: Tree network $N^T_{emb}$ with a set $F$ of dedicated selected nodes with its filter grammars (pre- and post-filters).
1: $G_F \leftarrow FilterGrammar(G, Q)$;
2: $v_{pre} \leftarrow ClosestPreFilterNode(N_{emb}, v_s, C, G_F)$;
3: $N^T_{emb} \leftarrow DisseminationTree(N_{emb}, v_{pre}, C)$;
4: $F \leftarrow PostFilterPlacement(N^T_{emb}, Q, v_{pre}, G_F)$;
5: $extendTreeByPreRoute(V^T, E^T, V, E, v_{pre})$;
6: return $\{N^T_{emb}, F\}$
```

called. This algorithm determines one processing node v_{pre} that is able to run the filter grammar G_F as well as that results in overall positive data dissemination. More precisely, we are not only considering the quality of the path to a processable node v_{pre} in terms of connection and device class, but also the quality from v_{pre} to all service subscribers.

2. Starting with the determined pre-filter node v_{pre} we discover an optimized dissemination tree N^T_{emb} from this v_{pre}. The *DisseminationTree* algorithm (line 3) will be called to gather such a tree. Thereby, relevant service data shall be delivered from v_{pre} to the service subscribers in a resource-optimized manner. More precisely, the data shall be routed via high quality connections, avoid very constrained embedded devices, and be shared for as long as possible if there are multi-client destinations. The latter can be fulfilled if one or more post-filters can be placed that retain the information of the final client destination nodes or the next post-filter nodes. Consequently, the *DisseminationTree* algorithm finds a dissemination tree from v_{pre} to all clients in C that takes into account the device class and connection quality metrics as well as the current processing capability of the potential post-filter placement.

3. Based on N^T_{emb}, suitable nodes are selected for the post-filter functionality to share service data as long as possible. The *PostFilterPlacement* algorithm (line 5) realizes this and provides the routing information for all filter grammars.

Before the *FilterEnabledDissemination* algorithm terminates, we extend N^T_{emb} by the involved nodes and connection that leads from v_s to v_{pre} (line 5).

For better clarification and to get an idea how our algorithm works we will consider an example in the next subsection applied on the network shown in Figure 1.

3.3 Example

We will now consider the embedded network $N_{emb} = (V, E, c, w, p)$ which is shown in Figure 1. Node 1 is a service provider and nodes 3, 4, and 7 are the service requesters with the following 4 query conditions as presented in Section 2.2:

Node 3: Q_1 and Q_2
Node 4: Q_3
Node 7: Q_4

Figure 3(b) already shows the filter grammar G_F based on this query set after applying the *FilterGrammar* procedure. Since the service requester node does not provide us with the opportunity to set up a filter mechanism for clients' subscription requests ($p(1) = 0$), we have to find an alternative node for placing a pre-filter. In order to do so, we have to identify any nodes with processing capabilities within the network that have enough resources to run G_F. The procedure *ClosestPreFilterNode* in Algorithm 1 will identify these nodes (3, 4, 5, 8) check their resource capabilities. If this results in more than one node, the node which is closest to the service provider and yields the best cost function value is selected based on test paths to clients in C using the Dijkstra algorithm (Dijkstra, 1959) with our metrics. The outcome would be node 5 that is selected as pre-filter node (v_{pre}) running G_F.
The next step involves determining a data dissemination tree that spans v_{pre} and client nodes 3, 4, and 7. In order to do so, we will call the *DisseminationTree* procedure. This algorithm is based on the concept of the Kou-Markowsky-Berman (KMB) algorithm (Kou et al., 1981) which is a well-known heuristic for the Steiner Tree problem. Based on our metric and cost model, respectively, Figure 4 shows the outcome of the *DisseminationTree* procedure.
The last major processing step in our dissemination algorithm involves determining suitable post-filter nodes to enable a high ratio of shared service data from service provider to service requesters. Starting with (root) node 5, the *PostFilterPlacement* procedure will first select all nodes that contain multi successor branches. Nodes 5 and 8 are candidates. Since node 5 already is a dedicated pre-filter node, we will not consider it further and instead check node 8 directly for processability of a post-filter grammar. The post-filter grammar is constructed based on the queries that can be reached from node 8. This is

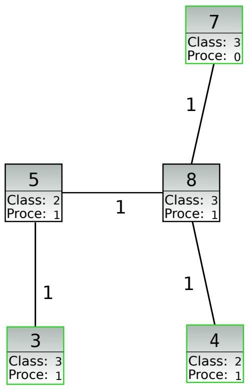


Figure 4: Determined tree based on the *DisseminationTree* procedure from v_{pre} (node 5) to all clients in C (nodes 3, 4, and 7).

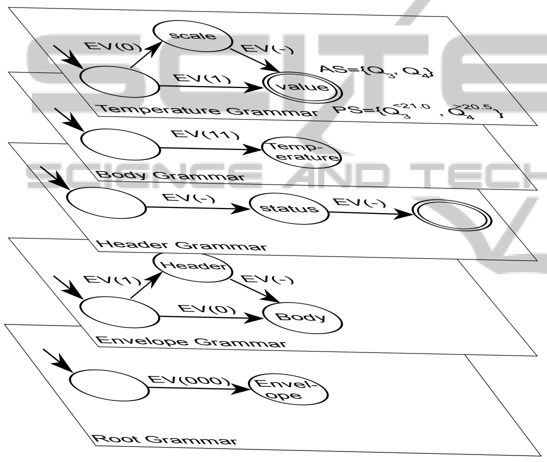


Figure 5: Post-filter grammar.

true for the queries Q_3 and Q_4. Figure 5 shows the post-filter as based on these queries that is constructed based on the mechanism presented in Section 2.2. Since node 8 is a node with processing capabilities we can successfully install the post-filter on this node.

Before the *PostFilterPlacement* algorithm is terminated, we are going to update the network's defined filters in terms of routing information. Node 5 is set up with the pre-filter G_F that is shown in Figure 3(b) and will receive all messages from node 1. In addition, node 5 with G_F contains the associated information Q_1 and Q_2 related to node 3, Q_3 related to node 4, and query Q_4 related to node 7. Based on the post-filter to be placed on node 8, service data that matches queries Q_3 and Q_4 shall be forwarded to node 8, which then will send the data only once. Thus, G_F is updated with this information. In summary, we obtain the following routing information:

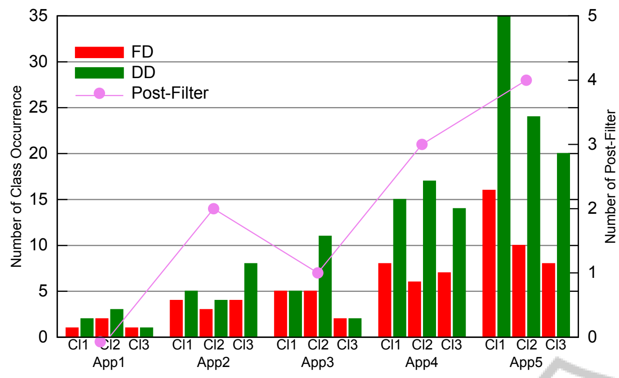
- G_F (at node 5): forwards service data to node 3 when queries Q_1 and/or Q_2 match; forwards service data to node 8 when queries Q_3 and/or Q_4 match.
- G'_F (at node 8): forwards service data to node 4 when Q_3 matches; forwards service data to node 7 when Q_4 matches

This determined data dissemination tree and the filter placement is also reflected in Figure 2(b).

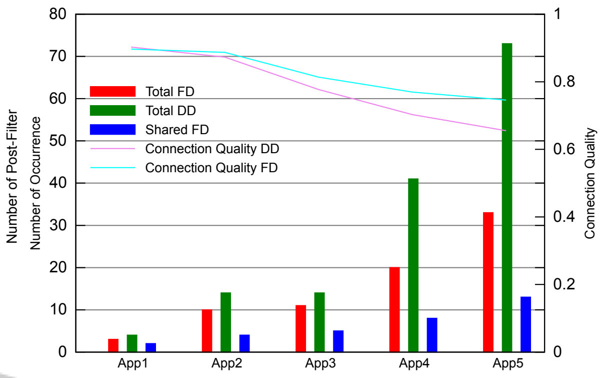
4 EVALUATION

So as to organize service data dissemination of each new applied application and to estimate its influence in terms of traffic and device capacity usage of real embedded networks we wrote an embedded network simulator. The simulator provides us with the opportunity to load particular network topologies and characteristics as well as service provider and the service subscribers with their queries. Another alternative is to setup randomized embedded networks by providing different kinds of generation parameters: number of nodes, number of different kinds of device classes, and the ratio of device classes and connection quality. Based on such a network, we are able to set up new applications by selecting particular nodes, which operate a service with the provided service description, and the client nodes that subscribe service data with the predefined conditions on the service data. We can then run our dissemination algorithm for each newly installed application.

In order to evaluate the effectiveness of the approach presented in this paper, we randomly generated an embedded network that has a complexity of 50 nodes with three device classes. This network setup initially features a balanced ratio of processable and non-processable nodes. Its class ratio consists of 5 times device classes 1, 10 times device classes 2, and 35 times device classes 3. Initially, we uniformly distributed the connection quality weighting values with numbers between 0.8 and 1. We sequentially installed five different kinds of applications. In general, an application is based on a service provider and different kinds of service requesters (the clients). The distance (in terms of hop count) and client distribution to the service provider node is increased with each new installed application. We start with the first application, which has two clients; subsequently, the second has 3 clients, the third has 4 clients, there are 5 clients in the fourth application, and finally the fifth application has 6 different service requesters. For each installed application we evaluated the service data dissemination for two variants: *Filter-enabled dissemination* (abbreviated with *FD*) represents our filter-enabled dissemination approach and the separate and *direct dissemination* (abbreviated with *DD*)



(a)



(b)

Figure 6: Embedded network with $|V| = 50$: (a) Count of used classes (Cl1, Cl2, Cl3) in the dissemination path and number of used post-filters for each application (App1, ..., App5). (b) Number of links of the dissemination path and average value of connection quality.

reflects the direct, non-filtered service data delivery (comparable with Figure 2(a)).

Figure 6 shows the evaluation results. Figure 6(a) depicts the result for each application in terms of device class occurrences (Cl1=Class 1 nodes, Cl2=Class 2 nodes, and Cl3=Class 3 nodes) in the dissemination path of our approach (FD Optimized) as compared to the simple approach, wherein each service data is delivered separately (DD Simple). In other words, we count the occurrence of the device classes in the determined dissemination path (tree) that reflects the worst case scenario when a service message is relevant for all service requesters in the network. As can be seen for all cases, our approach, as presented in this paper, results in a lower usage of class occurrences as compared to the simple service data distribution variant. This becomes especially apparent the more complex the application is. Furthermore, the occurrences also show that our determined dissemination paths always consist of the desirable, relatively small number of constrained nodes (class 3). For instance, in a worst case distribution scenario for application five, our dissemination approach uses the device class 1 sixteen times, class 2 ten times, and the most constrained device class 3 eight times. In total, 34 nodes are involved in the dissemination process. In contrast, a simple dissemination would lead to a device class ratio of class 1 thirty-five times, class 2 twenty-four times, and class 3 twenty times. In total, this involves 79 nodes. Consequently, our approach results in a better resource usage of the nodes in the embedded network since fewer total nodes are involved in the dissemination tree; the number of constrained nodes (class 3) is kept as small as possible.

Figure 6(b) shows the evaluation result in terms of the number of connection links used and average connection quality. As can be seen, the number of connections used in a dissemination process is smaller for our approach as compared to the simple variant. The figure also shows the ratio of the shared connections of the optimized variant in each application. We determined the number based on whether each connection between two nodes can reach a pre-filter or a post-filter node. If so, the number of shared connections is incremented. The presented numbers show the effectiveness of our approach, since for each application we determine a dissemination tree that consists of a high ratio of shared connections. Figure 6(b) also shows the average connection quality for each application and its dissemination based on both our approach and the simple variant. As can be observed, the simple dissemination variant loses the average connection quality faster than our approach. This is explained by the fact that the simple variant involves a lot more connection links and potentially causes more network traffic. The more applications that are installed in the network, the greater the impact on connection quality will be.

5 RELATED WORK

Finding a suitable pre-filter node outside of the service data origin node and the position of post-filters in a dissemination tree opens the opportunity to share relevant service data with a number of service subscribers. This leads to a reduction of resources used within embedded networks in terms of network traffic as well as processing overhead. Similar topics are addressed by and can be found in Data Stream Management Systems (DSMSs). DSMSs complement the traditional Database Management Systems (DBMSs). Typically, a DBMS handles persistent and random accessible data and executes volatile queries. Meanwhile, in DSMSs persistent (or long-running) queries are executed over volatile and sequential data. Examples of DSMSs include *Aurora* (Abadi et al., 2003), *Borealis* (Abadi et al., 2005), *TelegraphCQ* (Chandrasekaran et al., 2003), and *StreamGlobe* (Kuntschke et al., 2005). The main focus of such systems is on the efficient processing of potentially infinite data streams against a set of continuous queries. In contrast to publish/subscribe systems such as XFilter (Altinel and Franklin, 2000) or YFilter (Diao and Franklin, 2003), continuous queries in DSMSs can be far more complex than simple filter subscriptions. Some research developed new query languages such as WindowedXQuery (WXQuery) (Kuntschke, 2008) to extend query operations. In the domain of constrained embedded networks, however, we presume the presence of relatively simple data models and have found that XPath expressions are sufficient to address data interests and simple constraints by predicates. Other important work in distributed DSMSs such as StreamGlobe and Borealis revolves around network-aware stream processing and operator placement. These are issues also relevant to constrained embedded networks and, similarly, we took them into account for our approach by positioning the pre-filter and, if possible, the post-filter mechanism at the embedded nodes.

Most DSMSs, such as TelegraphCQ for example, are based on relational data. StreamGlobe, however, focuses on plain-text XML data streams as well as on XML-based query languages such as XQuery (Boag et al., 2007) or the above mentioned WXQuery. Consequently, nodes used for distributed data stream processing in systems such as StreamGlobe and Borealis generally need to be far more powerful than the microcontrollers for constrained embedded devices that we aim for in this paper. Our approach for constructing high performance filter mechanisms based on binary XML techniques enables us to bring DSMS topics to the domain of constrained embedded networks.

In our approach, filter nodes such as pre-filter or post-filter nodes decide how to best forward service messages if there are one or more matches. The destinations may include service requester nodes and/or other post-filter nodes. In the literature, this is called content-based routing or application-level routing since routing depends on the contents of data within a message. In that context, we can refer to works such as the *combined broadcast and content-based* (*CBCB*) routing scheme (Carzaniga and Wolf, 2002), the *application layer multicast algorithm* (*ALMA*) (Ge et al., 2006), the usage of *XML Router* (Snoeren et al., 2001), and *view selection for stream processing* based on XML data (Gupta et al., 2003). Below, we will concentrate on the latter since they also involve XML-based data content.

The XML Router approach (Snoeren et al., 2001) creates an overlay network that is implemented by multiple XML routers. An XML router is a node that receives XML packets and forwards a subset of these XML packets. The XML packets are forwarded to other routers or the final client node destinations. Thereby, the output links represent the XPath queries that describe the portion of the router's XML stream that should be sent to the host on that connection link. XML routers are comparable to our pre- and post-filter concept. However, additional strategies, such as reassembling a data packet stream from diverse senders provided by the diversity control protocol (DCP) or the usage of plain-text XML and XPath interpreters are not feasible in a resource constrained embedded environment.

The view selection for stream processing method is an interesting approach followed in (Gupta et al., 2003) and (Gupta et al., 2002). The main concept includes selecting a set of XPath expressions which are called *views*. The service data producers evaluate the views and add the result to the data package in the form of a header. The advantage is that servers which keep a local set of queries can evaluate their workload by inspecting only the values in the header and do not need to parse the XML document. This leads to a speed-up of routing decisions. However, this is only true for cases in which the evaluation in the header is positive. Otherwise, the complete (plain-text) XML document needs to be parsed and the query needs to be evaluated in a conventional way. Again, this is an obstacle in the constrained embedded environment. In addition, one of our goals is to achieve seamless protocol usage and to work with standardized message representations to support interoperability in a heterogeneous network environment. Adding a header to a message would break this principle and necessitate an adjustment of communication protocols.

6 CONCLUSION AND FUTURE WORK

In this paper, we presented an approach to realize efficient filter-enabled service data dissemination in constrained embedded networks based on XPath expressions given by different subscribers/clients. Finding a suitable pre-filter node in an embedded network leads to an early evaluation of relevant service messages. By using post-filters in a determined dissemination tree, we are able to avoid redundant transmissions and share the service data, especially if there is a multi-query match of different kinds of service requesters. The effectiveness in terms of device class occurrences, connection quality, and number of shared connections was demonstrated in a simulated environment based on our embedded network simulator.

Topics for future work include the dynamic update of client queries and their impact on the dissemination path as well as on the placed pre- and post filter grammar.

REFERENCES

Abadi, D. J., Ahmad, Y., Balazinska, M., Çetintemel, U., Cherniack, M., Hwang, J.-H., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., Tatbul, N., Xing, Y., and Zdonik, S. B. (2005). The design of the borealis stream processing engine. In *CIDR*, pages 277–289.

Abadi, D. J., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., and Zdonik, S. B. (2003). Aurora: a new model and architecture for data stream management. *VLDB J.*, 12(2):120–139.

Altinel, M. and Franklin, M. J. (2000). Efficient filtering of xml documents for selective dissemination of information. In Abbadi, A. E., Brodie, M. L., Chakravarthy, S., Dayal, U., Kamel, N., Schlageter, G., and Whang, K.-Y., editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 53–64. Morgan Kaufmann.

Boag, S., Chamberlin, D. D., Fernández, M. F., Florescu, D., Robie, J., and Siméon, J. (2007). XQuery 1.0: An XML query language. World Wide Web Consortium, Recommendation REC-xquery-20070123.

Bournez, C. (2009). Efficient XML interchange evaluation. W3C working draft, W3C. http://www.w3.org/TR/2009/WD-exi-evaluation-20090407.

Carzaniga, A. and Wolf, A. L. (2002). Content-based networking: A new communication infrastructure. *Lecture Notes in Computer Science*, 2538:59–??

Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M. J., Hellerstein, J. M., Hong, W., Krishnamurthy, S., Madden, S., Raman, V., Reiss, F., and Shah, M. A. (2003). Telegraphcq: Continuous dataflow processing for an uncertain world. In *CIDR*.

Diao, Y. and Franklin, M. J. (2003). High-performance xml filtering: An overview of yfilter. *IEEE Data Eng. Bull.*, 26(1):41–48.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

Ge, M., Krishnamurthy, S. V., and Faloutsos, M. (2006). Application versus network layer multicasting in ad hoc networks: the ALMA routing protocol. *Ad Hoc Networks*, 4(2):283–300.

Goldman, O. and Lenkov, D. (2005). XML binary characterization. World Wide Web Consortium, Note NOTE-xbc-characterization-20050331. http://www.w3.org/TR/2005/NOTE-xbc-characterization-20050331.

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., and Frystyk Nielsen, H. (2003). SOAP version 1.2 part 1: Messaging framework. World Wide Web Consortium, Recommendation REC-soap12-part1-20030624.

Gupta, A. K., Halevy, A. Y., and Suciu, D. (2002). View selection for stream processing. In *WebDB*, pages 83–88.

Gupta, K., Suciu, and Halevy (2003). The view selection problem for XML content based routing. In *PODS: 22th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*.

IEEE (2011). IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs). http://standards.ieee.org/about/get/802/802.15.html.

Käbisch, S., Kuntschke, R., Heuer, J., and Kosch, H. (2012). Efficient Filtering of Binary XML in Resource Restricted Embedded Networks. In *Proc. of the 8th International Conference on Web Information Systems and Technologies (WEBIST 2012)*, pages 174–182.

Käbisch, S., Peintner, D., Heuer, J., and Kosch, H. (2011). Optimized XML-based Web Service Generation for Service Communication in Restricted Embedded Environments. 16th IEEE International Conference on Emerging Technologies and Factory Automation.

Kou, Markowsky, and Berman (1981). A fast algorithm for steiner trees. *ACTAINF: Acta Informatica*, 15.

Kuntschke, R., Stegmaier, B., Kemper, A., and Reiser, A. (2005). Streamglobe: Processing and sharing data streams in grid-based p2p infrastructures. In Böhm, K., Jensen, C. S., Haas, L. M., Kersten, M. L., Larson, P.-Å., and Ooi, B. C., editors, *VLDB*, pages 1259–1262. ACM.

Kuntschke, R. B. (2008). *Network-aware optimization in distributed data stream management systems*. PhD thesis, Technical University Munich. urn:nbn:de:bvb:91-diss-20070806-625762-1-3; http://d-nb.info/989053113.

Schneider, J. and Kamiya, T. (2011). Efficient XML Interchange (EXI) Format 1.0. http://www.w3.org/TR/exi. W3C Recommendation 10 March 2011.

Snoeren, A. C., Conley, K., and Gifford, D. K. (2001). Mesh based content routing using XML. In *SOSP*, pages 160–173.