Bandwidth-Minimized Distribution of Measurements in Global Sensor Networks

Andreas Benzing, Boris Koldehofe, and Kurt Rothermel

Institute of Parallel and Distributed Systems, 70569 Stuttgart, Germany
firstname.lastname@ipvs.uni-stuttgart.de
http://www.ipvs.uni-stuttgart.de/abteilungen/vs

Abstract. Global sensor networks (GSN) allow applications to integrate huge amounts of data using real-time streams from virtually anywhere. Queries to a GSN offer many degrees of freedom, e.g. the resolution and the geographic origin of data, and scaling optimization of data streams to many applications is highly challenging. Existing solutions hence either limit the flexibility with additional constraints or ignore the characteristics of sensor streams where data points are produced synchronously.

In this paper, we present a new approach to bandwidth-minimized distribution of real-time sensor streams in a GSN. Using a distributed index structure, we partition queries for bandwidth management and quickly identify overlapping queries. Based on this information, our relay strategy determines an optimized distribution structure which minimizes traffic while being adaptive to changing conditions. Simulations show that total traffic and user perceived delay can be reduced by more than 50%.

Keywords: Data Streams, Global Sensor Networks, Optimization.

1 Introduction

The deployment of wired and wireless sensor networks is making rapid progress all around the globe. Sensors are installed in virtually all areas of everyday life and a huge amount of real-time data has become available. For example, the CeNSE project by HP [12] envisions one trillion sensors to enervate the entire Earth. To globally enable scalable access to this data, broker networks providing high bandwidth data streams need to be established and managed using appropriate middleware solutions. Previous approaches focused on extending methods from local to global sensor networks [2, 13]. By utilizing in-network aggregation or lossy compression, a significant reduction of sensor data can be achieved. However, the gain of these methods comes at the cost of reduced precision to which many critical applications are highly sensitive. For example, the simulations for the dispersion of pollutants require precise wind data. Numerical errors due to compression artifacts can lead to highly disturbed results.

K. Magoutis and P. Pietzuch (Eds.): DAIS 2014, LNCS 8460, pp. 156–170, 2014.

A promising alternative to reducing precision is to avoid sending information multiple times over a physical network link. The main problem then is to find an optimized and dynamically adapting dissemination structure for a large number of requests in the complex Internet. Group communication middleware such as application layer multicast [24] or publish/subscribe [9] tackle the complexity of the problem by reducing the number of requests which need to be served by a broker. In multicast, communication is limited to a fixed number of channels, prohibitively restricting the expressiveness of queries. Publish/subscribe offers high expressiveness for queries and scalability is achieved by merging similar subscriptions from different subscribers to avoid redundant transmissions. However, since the bandwidth requirements of an individual subscription are unknown, this solution misses optimization potential for high-bandwidth sensor streams.

Contrary, for global sensor streams, measurements are generated for every point at discrete time steps. Knowledge on the spatial and temporal resolution – as specified, for example, in scientific simulations – allows for exact determination of the data to expect from any given query. This poses a great so far unused potential to minimize bandwidth usage of a global sensing system.

Towards this end, we present the Global Sensor Grid Middleware (GSGM). The GSGM provides a new method to precisely query and scalably deliver real-time sensor data streams. The query interface allows for a flexible specification of geographic regions and resolutions of data of interest. Our main contribution is a new underlay-adaptive stream management algorithm to minimize traffic by exploiting the predictable network load of each query. Redundant parts of queries are quickly identified using a distributed index structure. Identical portions of data streams are merged and sent to dynamically established relay points where the data is distributed towards client nodes based on local knowledge.

The paper is structured as follows: Section 2 introduces the system model followed by a problem formalization in Section 3. The detailed description of our approach is provided in Section 4. Evaluation results are presented in Section 5. Related work is provided in Section 6 and Section 7 concludes the paper.

2 System Model

System Components. The GSG is built up of *brokers*, which gather data from sensor networks, and *clients*. The sensor networks expose coordinates of their nodes and send raw sensor measurements to brokers where they are pre-processed and distributed to interested clients as shown in Figure 1. In doing so clients can pose a query to a broker, identifying the type of sensor, the spatial region, and spatial as well as temporal resolution. The brokers in turn identify the relevant data and establish a continuous sensor stream to the clients. In order to access the sensor data provided by a sensor network, we assume the sensor networks provide specific interfaces to the GSGM. These interfaces allow querying sensor data at varying temporal dimensions as well as coordinates in a global coordinate system. Note, however, that the GSGM is not concerned with managing the internals of the sensor network and how the sensor network is established.

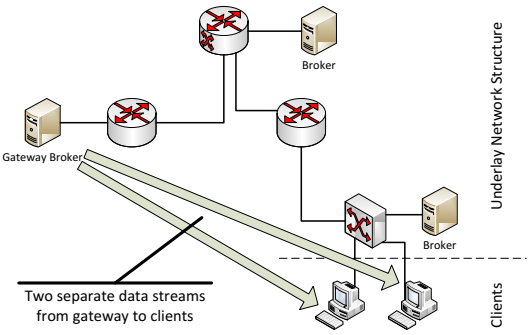

Fig. 1. Naive operation of the GSG with a separate data stream per client

To uniformly manage the sensors and sensor networks connected to the brokers, the GSGM manages a global coordinate system which comprises the world of all possible sensor coordinates. This world is partitioned into disjoint geographic regions, each managed by a single broker denoted the *gateway* of said region. This way each sensor is associated with exactly one region. However, a broker may serve as gateway for multiple regions. The gateways perform preprocessing of the raw sensor data to integrate sensor data into the global coordinate system and being able to offer sensor data at full resolution (cf. [3]).

Additional brokers in the GSG support the distribution of sensor streams. We refer to brokers that forward sensor data on behalf of gateways as *relays*. Note that also gateways may serve as relays on behalf of other gateways. Gateways or relays, which send data to a client, form the end of a data stream inside the GSG and hence called *target brokers* of that client. The number of brokers as well as the assignment of gateways to regions is manually fixed by the operator of the Global Sensor Grid. This choice determines the number of queries as well as the performance characteristics that can be supported by the GSGM. However, the brokers will autonomously optimize the network resources used depending on the client requests for sensor data.

Query Model. Building on our previous work [4] we introduce the query model for the GSGM. A query to the GSGM is bound to one data type and includes the region of the data requested. Optionally, clients can specify the time of the beginning and duration of the data stream. Each query q is therefore given by a tuple $q = (R_q, t_q, d_q)$ where R_q specifies the geographical extent and resolution of the query. t_q and d_q denote the optional start time and duration of the query. If the latter two are not specified, the resulting data stream will start as soon as possible and run until the client fails or cancels the query.

Besides the geographical area of interest, the query region contains the requested resolution. This way, clients can directly query a grid of data points as required for the simulation at hand. The region of query q is represented by

$$R_q = (x_{min}, x_{max}, y_{min}, y_{max}, res_x, res_y, res_t),$$

where x_{min} and x_{max} describe the lower and upper bound of the latitude. y_{min} and y_{max} limit the longitude correspondingly. Note that the system is not limited to geographic coordinates but can process any 2D coordinates. The last three values res_x, res_y, and res_t specify the spatial and temporal resolution of the query as a fraction of the maximum available data. A value of $res_x = 0.3$ therefore returns 3 out of 10 data points equally distributed over the corresponding axis. Similarly, only a subset of updates is transmitted for each data point if $res_y < 1$ or $res_t < 1$.

Cost Model. The primary goal of the GSGM is to minimize the bandwidth used for distributing data. The bandwidth used by each data stream is proportional to the amount of data per update and the update frequency. On the one hand, the amount of data is given by the geographical extent of the area and the spatial resolution. On the other hand, the frequency depends on the temporal resolution. For a data stream serving a query q the corresponding total size $|q|$ is therefore given by:

$$|q| = (x_{max} - x_{min}) \times (y_{max} - y_{min}) \times res_x \times res_y \times res_t$$

While the size of a query determines the bandwidth required to send the data from a broker, the load on the network depends on how data is distributed in the network. To serve a query q, the corresponding data stream s_q uses a set L_{s_q} of underlay links, which depends on the chosen path in the network. The number of underlay links for this path is consequently denoted by $|L_{s_q}|$. The overall cost $c(s_q)$ can now be calculated as the product of the number of links and the bandwidth which is identical for each link: $|L_{s_q}| \times |q|$. Note that the delay of a link is not taken into account as it does not affect the load.

3 Sensor Stream Distribution Problem

Recall that the bandwidth usage of the GSGM is driven by the overall number of underlay hops over which a given sensor datum is forwarded. To achieve minimal bandwidth usage it is therefore important to i) exploit the overlap between distinct sensor streams to reduce the amount of sensor data which is forwarded, ii) carefully choose the brokers which contribute in relaying to ensure short underlay paths. These goals conflict in general since longer underlay paths are required to share an underlay link between multiple streams.

We focus on two interconnected sub-problems to find a distribution structure with minimized bandwidth usage. The first sub-problem is to efficiently identify *intersections* between distinct queries. For each intersection, the data covered by the overlapping region must be sent to multiple clients. Solving the problem allows us to identify a set of k non-overlapping sensor streams from a single gateway to possibly multiple clients. The second task, and main problem addressed in this paper, is to reduce the bandwidth required for each stream by finding a minimal cost distribution graph. Clearly, both problems are in general hard to solve: First, the amount of possible intersections grows quickly with the number

of queries in the system. Second, the path distribution problem can be reduced to the Minimal-Steiner-Tree-Problem (MST) [8] which is known to be NP-hard.

Proof. We can formulate this optimization problem similar to the MST w.r.t. each intersection. In the generic version of the problem, we are given a graph $G = (V, E)$, one source node $s \in V$, and a set $R \subseteq V$ of required nodes. The sought-after Steiner Tree is the graph which connects s to all nodes in R using nodes in $V \setminus R$ as Steiner Points, i.e. relays, with minimal sum of edge weights.

To formulate our stream distribution problem for a single intersection, let $V = \{b_i\}_{i \in \{1,\ldots,n\}}$ be the set of brokers, b_s be the source broker which provides the data, and R be the target brokers to which the clients are connected. Since a direct connection between any broker $b_i \in V$ can be established using IP-routing, The solution to this problem is the desired tree which provides the connectivity at minimal cost.

Even though multiple approaches to the MST exist, finding the exact solution to the presented problem is infeasible in a real-world GSG for multiple reasons. First of all, the MST is an NP-hard problem and has to be solved for a very large graph, i.e. the Internet with all its links. A restriction to all direct connections between brokers still involves a an extremely large graph. Second, a solution is needed for every intersection, resulting in numerous instances of the problem. Finally, the topology of the Internet is not known. This knowledge gap is broadened by the continuously changing network conditions caused by other applications using the Internet. In the remainder of this paper, we therefore present a new approach to solving the sensor stream distribution problem.

4 Sensor Stream Organization

In this section, we describe our new approach to approximate a solution to the sensor stream distribution problem and show how data streams in the GSG are organized to minimize bandwidth consumption. An example is given in Figure 2 The approach is divided into four main parts: query processing, management of network information, relay selection algorithm, and merging algorithm.

Query Processing. The query processing framework builds on our previous work [4] and provides three functions: i) find the gateway for a given query, ii) identify the overlap of distinct queries, and iii) determine which queries to serve with incoming or locally generated data. We use the GBD-Tree [16] which is able to index very large, high dimensional spatial data sets. In our case, each broker maintains a GBD-tree to store queries according to their regions as well as gateways by the regions they serve. This way, a broker can determine which parts of a query can be served locally and where to forward other parts.

Each region in the GBD-Tree is labeled by a binary string called a DZ expression as illustrated in Figure 3. The DZ expression exposes two important properties: First, the shorter a DZ expression, the larger is the extent of the

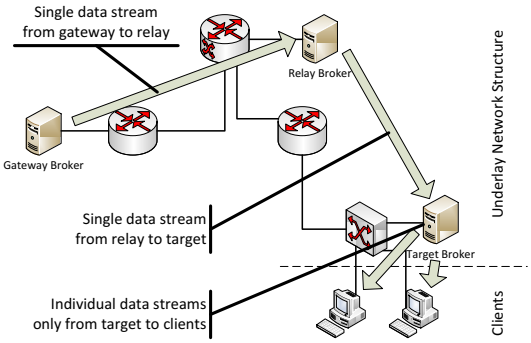

Fig. 2. Merged data streams using a relay broker in the network. A single data stream is sent over the relay to the target broker which then serves both clients.

region. In particular, the empty DZ expression ϵ corresponds to the entire coordinate space of the GSGM. Second, if the DZ expression of region r is the prefix of another region r' then r' is fully contained in r. For example, regions 010 and 011 both reside completely inside region 01. The mapping of DZ expressions to a region can be efficiently achieved by means of recursive spatial decomposition.

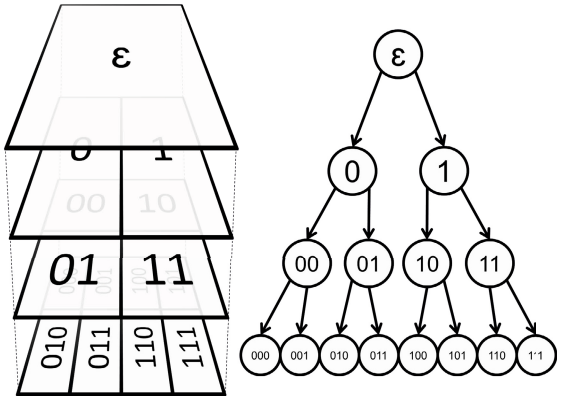

Fig. 3. Example for DZ expressions generated by spatial indexing

To determine the gateways for a query, the region of a query is converted to a set of DZ expressions according to the illustrated spatial decomposition scheme. As the tree is traversed top-down, the nodes will eventually contain information about the gateway on which the sensor data for the corresponding region is located. Similarly, for locally generated data or an incoming stream, the queries which are served by that stream can be found. By traversing the tree in a bottom-up fashion, we can aggregate information about overlapping queries as detailed in Section 2. Note that we only used two-dimensions in the example

for clarity. The scheme supports various spatial and temporal dimensions. As queries are dynamically relayed to remote brokers, additional state is maintained by the GSGM. For each incoming relay stream, the path from the corresponding gateway to the local broker is stored in the index structure to prevent circular relaying. Each query is additionally annotated with the corresponding target broker to allow for efficient processing on the relay broker.

Neighborhood Maintenance. Direct connections, i.e. IP routing without any relay points in between, provide fast routing of queries and low delay in responses. However, for increasing load, multiple streams might carry redundant data over the same underlay link. These similar data streams are therefore *merged* and only a single stream will be forwarded to a selected relay point. As described in Section 3, selecting a relay point requires information about the underlying network.

Each broker maintains a local view on the network, to infer the structure of the underlying topology. This view contains a limited number of topographically close brokers, called local neighborhood and additional information about each of these brokers. Initially, all the brokers from the information about the region assignment are added to the neighborhood. The neighborhood is then continuously updated with new candidates during query routing and neighborhood maintenance itself. When a broker learns about another, from the region assignment or through an update, the remote broker is added to the neighborhood and delay measurements to the new broker are initiated. The delay is used to estimate the topological distance to each of the remote brokers which a broker uses to decide which remote brokers to keep in the neighborhood. Close brokers are kept in the neighborhood while brokers with higher delay are removed from the neighborhood and measurements to them are stopped. However, to adapt to changing network conditions, the brokers which are currently not in the neighborhood are periodically measured. No additional topology information, e.g. data from network oracles, is used by the middleware.

To assess the network location of a remote broker, the local view contains additional information. After a broker is added to the local neighborhood, the neighborhood of that new broker is requested together with the respective measurements. This way, each broker can check the network from the point of view of any of the brokers in its neighborhood, resulting in a two-hop view. Brokers in the remote neighborhood that were previously unknown are considered for the local neighborhood as previously described.

Relay Selection. In this section, we describe our approach to relay selection which serves as the basis for merging data streams. Relay selection is executed locally on each broker to determine where to send relay data streams and which clients to serve directly. In the following, we first introduce our notion of data stream similarity to provide an intuitive optimization goal. We then work towards our strategy for selecting relay candidates accordingly.

Basic Principles Merging overlapping data streams will only reduce the bandwidth consumption if they share at least part of an underlay path. Although the query index allows us to efficiently identify overlapping queries, we do not yet know whether the corresponding streams share any links in the underlay path. We approach this problem by introducing a similarity metric to provide an intuition about the problem at hand and a qualitative measure whether or not it is worth merging two streams.

For the GSGM, the similarity of two overlapping data streams is determined by the fraction of underlay links which they have in common in relation to the total path length. Formally, given two data streams s_1 and s_2 which carry the same data on paths p_1 and p_2 respectively, where $|p_i|$ denotes the length of path i and $l_{i,j}$ denotes the j^{th} link on path i, the similarity is given by:

$$sim(p_1, p_2) = \sum_{l_{1,k}=l_{2,l}} \frac{2}{(|p_1| + |p_2|)}$$

If the paths are identical, $|p_1| = |p_2|$ links will be summed up, resulting in a similarity of one. If the paths have no links in common, the similarity is zero.

Note that calculating the exact value of similarity requires knowledge of the entire underlay network. However, as described previously, the network graph is neither known, nor can it be measured exactly. Instead of first calculating the similarity, we therefore directly approach the relay selection problem in the GSGM. Relay selection is executed locally for each known target broker since their number is small compared to the large count of intersections. A solution to the problem ensures two properties: a) the network stretch is minimized, i.e. the path including the relay is as short as possible, while b) the probability of selecting the same relay for similar streams is maximized. The path segment from the originating broker to the selected relay can then be merged, i.e. served by a single stream, to reduce the load on the network.

Analysis of Simple Relay Strategies. The simplest relay strategy is to forward data streams directly to a client's target broker. We refer to this as *direct relay* strategy. Since the target broker is chosen by location close to the client in our setup, the introduction of the relay will result in a small network stretch. At the same time the strategy allows to remove all redundant transmissions with respect to clients which have overlapping queries and the same target broker.

Clearly, selecting additional relays between a gateway and the target brokers will increase the chance of multiple overlapping streams on the same relay link. At the same time, however, additional relays increase network stretch resulting in additionally used underlay links and higher overall bandwidth usage. In general, relying on the local neighborhood allows us to limit the stretch introduced by each additional relay. Furthermore, limiting the set of relay candidates increases the chance to find a common relay point for overlapping data streams.

This basic reasoning results in a simple greedy strategy, *closest to destination*. To establish the relay path, the gateway will initially identify the relay which is topologically closest to the client. Each established relay broker, will then do the

same until the target broker has been reached and the data is forwarded to the clients. While this greedy approach achieves low network stretch, its drawback is the large number of different relays chosen. As a result, the merging algorithm can only join streams which already have a very high similarity and the traffic reduction properties of this approach are sub-optimal.

To exploit lower similarity of data streams by having more relays, the *closest to source* strategy selects the relay based on the distance to the originating broker. First, the neighbors are sorted by increasing distance to the originating broker. Then, the first candidate which is closer to the target broker is chosen as relay. Note that this strategy is very fast, independent of the size of the neighborhood. As a result, the data streams are routed over a relatively large number of hops. While this provides the possibility to merge small common path segments, the network stretch grows very quickly.

```
Data: dst: Destination Broker, neighbors: Neighbors Sorted by Distance
Result: relay: Relay Broker
oldDist = MAX_DIST;
relay = dst;
for n ∈ neighbors do
    if getDelay(n, dst) > getDelay(this, dst) then
        continue;
    end
    newDist = getDelay(n, dst) + getDelay(this, n);
    if newDist < oldDist then
        oldDist = newDist;
        relay = n;
    end
end
```

Algorithm 1. Closest to Path Relay Selection

Combined Strategy for Minimized Usage We conclude the previous analysis with the relay selection strategy used in the GSGM. Rather than selecting a position relative to the source or target of a data stream, the relay is chosen *closest to the path* between the two. The pseudo code for the strategy is given in Algorithm 1. Before the relay selection is started, the brokers in the neighborhood are sorted with increasing distance to the source broker. For each candidate we first check whether it is closer to the target broker than the direct connection to avoid detours. Then we estimate the network stretch each suitable relay candidate. The broker with the least estimated distance to the direct path is kept as the result of the relay selection. If no suitable relay broker is found in the local neighborhood, the target broker is selected as relay.

We gain in two aspects towards our goal. First, the distance to the next relay is limited not only by the neighborhood but by selecting closer brokers with higher priority. This allows for an increased number of relays compared to the

greedy closest to destination strategy. Since we ensure that relays are located in the direction to the respective targets, similar streams will be merged with high probability. Second, the network stretch is limited as the relay nodes lie close to the direct path towards the target brokers. By selecting the relays with possibly low network stretch, we limit the number of underlay links used in the path. Data is therefore delivered with low delay and bandwidth usage is reduced.

Merging Algorithm. The relay selection provides a relay for each target broker and therefore for each data stream. Based on the selection, each broker has to locally select which streams to merge towards each relay. The goal of the merging operation is to find possibly large intersections for relaying to merge as many streams as possible. A basic approach for this operation is to find large queries which carry a superset of data of smaller queries. The query index provides the foundation for quickly identifying this containment of distinct queries. Our novel merging algorithm furthermore exploits the structure to split queries in order to increase the overlap and therefore increase efficiency and flexibility. A parameter thereby limits extreme fragmentation to ensure a low overhead.

The selection and adaption of regions works in two main steps. First, additional information in the index is gathered. A query q is contained in the sub-tree marked by node n if the region assigned to n covers the region of q denoted by $R_q \leq R_n$. In addition, we use the result of the relay selection to group queries by their relay candidate. For a target broker t_q of query q the relay is given by $relay(t_q) = r$. The set of queries for a node n and relay r is now given by $Q_{n,r} = \{q : R_q \leq R_n \wedge relay(t_q) = r\}$. The total size of queries per node n and relay broker r can then be described by the total size of queries in $Q_{n,r}$:

$$size(n, r) = \sum_{q \in Q_{n,r}} |q|$$

Recall that the size of queries is proportional to the bandwidth required by them. By calculating $size(n, r)$ for all nodes and relay candidates, we annotate the index with the size of queries per relay candidate in each node. Note that this operation can be computed very efficiently as query sizes can be aggregated by a bottom-up traversal of the index.

In the second step, the query index is traversed top-down for nodes containing at least one query by recursively calling `mergeStreams` as described in Algorithm 2. If a node contains more than one query which can be relayed by the same broker, a new relay is established, the corresponding region is added to R. All matching queries in the sub-tree, including any existing relay streams, are merged into that relay, i.e. added to the set Q of queries to forward. If there is only a single query for the currently considered relay and $size(n, relay) > R_n + \Delta$, the query is split and the two parts are added to the two corresponding child nodes; the process then continues on each of the two child nodes. Δ is a system parameter which controls how small queries are split to limit fragmentation.

To establish the relay data stream, all affected queries are forwarded to the relay. A copy of the queries is kept locally for future merging attempts. Once all

```
Data: n: Index Node, relay: Relay Broker
Result: Q: Set of Queries to Forward, R: Set of Regions to Stream to Relay
if getQueries(n, relay) == 0 ∧ size(n, relay) > 2Δ then
    for child ∈ getChildren(n) do
        mergeStreams(child, relay);
    end
end
if getQueries(n, relay) >= 2 then
    Q = Q ∪ getUnassigned(n, relay);
    R = R ∪ getRegion(n);
else
    if size(n, relay) > R_n + Δ then
        pushToChildren(getQueries(n, relay));
        for child ∈ getChildren(n) do
            mergeStreams(child, relay);
        end
    end
end
```

Algorithm 2. Stream Merging Algorithm

queries have been sent to the relay broker, the source broker signals the relay broker the end of the relay step. The single data stream containing the requested region is then sent to the relay broker and the relay broker starts relaying queries.

5 Evaluation

The GSGM has been implemented using the OMNeT++ network simulator [23] with the inet framework for IP-traffic simulation. A realistic topology was generated using the tools of ReaSE [10]. The topology consists of seven autonomous systems, two of which are set up as transit domains. The AS themselves have a hierarchical structure, based on up to three core routers on the top level which are fully connected. The second level consists of up to three gateway routers per core which are connected to the edge routers on the third level. Client nodes are connected to the edge routers using asymmetric DSL. Brokers are placed depending on the scenario. The network setup included 577 clients and an average of 48 brokers where the brokers where directly connected to routers on different levels. Clients generated queries in uniformly distributed time intervals between 30s and 300s. Each query lasted between 60s and 250s. The total simulated time span was 20 minutes. The queried regions were distributed around a single point with exponentially decreasing probability of larger distance to simulate peak load on a region. Since the location of brokers in the networks is crucial for the performance of the overall system we varied the broker placement. For the core placement, Brokers are either placed in data centers at the core of the network and connected to gateway and core routers or placed at the edge to simulate sensor gateways executing the GSGM. The full placement comprises brokers

distributed randomly among all routers. We measured the data transmitted by core and gateway routers only since the links to brokers are dedicated to the GSG and the links to clients cannot be optimized.

All evaluations include the native operation (OFF). As benchmark setups, we chose to add the simple strategies presented in Section 4, including DIRECT relay only, closest to destination (CTD), and closest to source (CTS). In the farthest from source (FFS) strategy a relay is selected farthest from the source broker. By prioritizing the distance to source, the strategies result in higher network stretch and therefore diversity in paths for load balancing. Our newly developed closest to path strategy is abbreviated CTP. The numbers attached to the label show the corresponding neighborhood size, where applicable. Overall, all relay selection strategies perform well considering the total traffic used by the system. Note that by selecting a relay closest to the path (CTP) and restricting the neighborhood, the data transmission is reduced most in all runs.

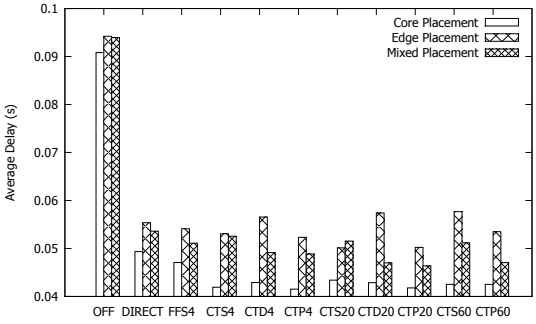

Fig. 4. Average user perceived delay for core placed brokers

The reduced bandwidth usage of our system has a positive effect on the quality of service. In a high bandwidth streaming system, the user perceived delay is significantly caused by queuing. By distributing fewer data on a single broker and leveraging more relay nodes, a larger neighborhood is beneficial for reducing the client perceived delay. However, the additional processing and propagation delay leads to increased client perceived delay for many overlay hops. For direct relaying, the many outgoing data streams on the gateway broker lead to increased queuing delay, while the prohibitively large network stretch of the farthest from source strategy increases propagation delay even for small neighborhood sizes. The trade-off can be seen in Figure 4 which also shows that closest to path (CTP) also performs best for this metric.

To investigate capacity improvement, queries were added to the system for a single gateway broker distributed over five minutes. We measured the degradation in QoS over time to show the gain in overall scalability. As Figure 5 shows, the system is able to serve all queries with stream merging enabled whereas without the delay quickly becomes prohibitively high. Note that the size of client queries is limited in all cases by the bandwidth of links to the client to avoid any effects from queuing delay and packet drops on the last mile.

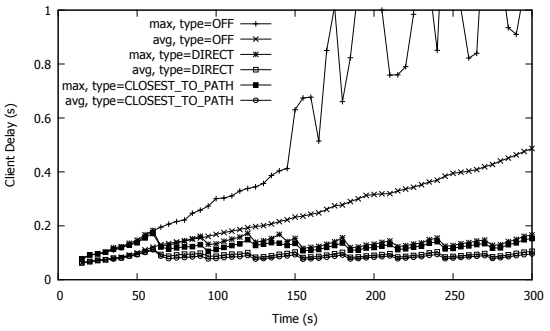

Fig. 5. Client perceived delay with increased load over time for core placed brokers

6 Related Work

From an architectural point of view, the GSGM is related to the OGC sensor web enablement (SWE) framework [18]. However, the main goal of the SWE is to contact single sensors rather than providing data streams for scientific simulations. While DataCutter [5] provides an interface similar to the GSGM and extended filter functionality, it is not optimized for real-time data.

In the field of data reduction in networks, three categories have been addressed so far: The first category, application layer multicast [15, 24] has been proposed to handle redundant queries for identical data. These systems are tailored for channel-based communication, where a limited number of channels needs to be served to a high number of clients. Consequently, they rely on the expensive detailed generation of a distribution tree. However, the content of sensor data streams is determined by individual queries rather than channels. Although extensions have been proposed to map queries to channels [7], the resulting false positives incur prohibitively high load on the network. Content-based pub/sub systems [9] as the second category aim to establish a containment relationship between the data sources and sinks, i.e. sensors and clients. They provide a flexible subscription interface which is capable of handling arbitrary queries. Pub/sub systems typically use clustering [21] or merging [6, 14] of subscriptions to reduce the processing overhead at the cost of increased bandwidth usage. Recent approaches also consider the physical structure of the network [22]. However, events occur rarely compared to sensor data which is sent out in high volume for all queried regions. Therefore, these approaches are not suitable to serve arbitrarily intersecting queries for sensor data and their resulting high bandwidth data streams. A third related category, stream processing systems [1, 11, 17, 19, 20], move processing tasks to optimize application performance and minimize network usage. Based on the underlying assumption that some of the processing tasks which can be moved reduce the bandwidth, these tasks are placed closer to the data sources. The remaining tasks are placed in a way to minimize the amount of data that is currently in transit on the network. However, the goal of

these systems is to optimize latency in presence of complex interaction graphs rather than to minimize total bandwidth usage.

7 Conclusion

In this paper we presented our approach to minimize bandwidth usage for measurement distribution in a Global Sensor Grid. Using a distributed index, we partitioned queries for bandwidth management and could efficiently identify intersections between queries. Based on this knowledge, our relay strategy built an optimized distribution structure for data streams. Redundant streams are thereby joined by our new merging algorithm. The approach has been evaluated on an Internet like network topology. The results show that our optimized relay strategy yields at least 33% reduction in overall traffic. Furthermore, our approach can fully exploit brokers placed in the core of the network, achieving a total reduction of traffic of over 50%. The evaluations also show the positive effects on user perceived delay. In the future, we aim to combine the relay strategy with a clustering algorithm to better control the distribution structure.

Acknowledgments. The authors A.B. and K.R. would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

References

1. Abadi, D., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J., Lindner, W., Maskey, A., Rasin, A., Ryvkina, E., et al.: The design of the borealis stream processing engine. In: CIDR 2005 (January 2005)
2. Ahmad, Y., Nath, S.: Colr-tree: Communication-efficient spatio-temporal indexing for a sensor data web portal. In: ICDE 2008, pp. 784–793 (April 2008)
3. Balazinska, M., Deshpande, A., Franklin, M., Gibbons, P., Gray, J., Nath, S., Hansen, M., Liebhold, M., Szalay, A., Tao, V.: Data management in the worldwide sensor web. IEEE Pervasive Computing 6(2), 30–40 (2007)
4. Benzing, A., Koldehofe, B., Rothermel, K.: Efficient support for multi-resolution queries in global sensor networks. In: Proc. of the 5th International Conference on Communication System Software and Middleware, COMSWARE 2011, pp. 11:1–11:12. ACM, New York (2011)
5. Beynon, M.D., Kurc, T., Catalyurek, U., Chang, C., Sussman, A., Saltz, J.: Distributed processing of very large datasets with datacutter. Parallel Computing 27(11), 1457–1478 (2001)
6. Bianchi, S., Felber, P., Gradinariu, M.: Content-Based Publish/Subscribe Using Distributed R-Trees. In: Kermarrec, A.-M., Bougé, L., Priol, T. (eds.) Euro-Par 2007. LNCS, vol. 4641, pp. 537–548. Springer, Heidelberg (2007)
7. Boukerche, A., Roy, A., Thomas, N.: Dynamic grid-based multicast group assignment in data distribution management. In: DS-RT 2000: Proceedings of the Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications, pp. 47–54 (2000)

8. Dreyfus, S.E., Wagner, R.A.: The steiner problem in graphs. Networks 1(3), 195–207 (1971)
9. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/subscribe. ACM Computing Surveys 35(2), 114–131 (2003)
10. Gamer, T., Scharf, M.: Realistic simulation environments for IP-based networks. In: Simutools 2008, pp. 83:1–83:7. ICST (2008)
11. Gibbons, P., Karp, B., Ke, Y., Nath, S., Seshan, S.: Irisnet: An architecture for a worldwide sensor web. IEEE Pervasive Computing 2(4), 22–33 (2003)
12. Hartwell, P.: Cense: A central nervous system for the earth. In: 2011 IEEE Technology Time Machine Symposium on Technologies Beyond 2020, p. 1 (2011)
13. Iwanicki, K., van Steen, M.: Using area hierarchy for multi-resolution storage and search in large wireless sensor networks. In: ICC 2009, June 14-18, pp. 1–6 (2009)
14. Jayaram, K.R., Jayalath, C., Eugster, P.: Parametric subscriptions for content-based publish/subscribe networks. In: Gupta, I., Mascolo, C. (eds.) Middleware 2010. LNCS, vol. 6452, pp. 128–147. Springer, Heidelberg (2010)
15. Kurian, J., Sarac, K.: A survey on the design, applications, and enhancements of application-layer overlay networks. ACM Comput. Surv. 43(1), 5:1–5:34 (2010)
16. Ohsawa, Y., Sakauchi, M.: A new tree type data structure with homogeneous nodes suitable for a very large spatial database. In: ICDE 1990, February 5-9, pp. 296–303 (1990)
17. Pietzuch, P., Ledlie, J., Shneidman, J., Roussopoulos, M., Welsh, M., Seltzer, M.: Network-aware operator placement for stream-processing systems. In: ICDE 2006, p. 49. IEEE (2006)
18. Reed, C., Botts, M., Davidson, J., Percivall, G.: OGC ® sensor web enablement: Overview and high level achhitecture. In: 2007 IEEE Autotestcon, pp. 372–380 (2007)
19. Rizou, S., Dürr, F., Rothermel, K.: Fulfilling end-to-end latency constraints in large-scale streaming environments. In: IPCCC 2011, pp. 1–8 (November 2011)
20. Srivastava, U., Munagala, K., Widom, J.: Operator placement for in-network stream query processing. In: PODS 2005, pp. 250–258. ACM (2005)
21. Tariq, M.A., Koldehofe, B., Koch, G.G., Rothermel, K.: Distributed spectral cluster management: A method for building dynamic publish/subscribe systems. In: DEBS 2012: Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, pp. 213–224. ACM (2012)
22. Tariq, M.A., Koldehofe, B., Rothermel, K.: Efficient content-based routing with network topology inference. In: DEBS 2013: Proceedings of the 7th ACM International Conference on Distributed Event-based Systems, DEBS 2013, pp. 51–62. ACM, New York (2013)
23. Varga, A.: Omnet++. In: Wehrle, K., Güneş, M., Gross, J. (eds.) Modeling and Tools for Network Simulation, pp. 35–59. Springer (2010)
24. Yeo, C., Lee, B., Er, M.: A survey of application level multicast techniques. Comp. Comm. 27(15), 1547–1568 (2004)