QUERY ROUTING OPTIMIZATION IN SENSOR COMMUNICATION NETWORKS

Guofei Jiang and George Cybenko
Institute for Security Technology Studies and
Thayer School of Engineering
Dartmouth College, Hanover NH 03755

Abstract

Interest in large-scale sensor networks for both civilian and military applications is burgeoning. The deployment of such networks will require new approaches in resource discovery, query processing and data routing. This paper presents a framework and some analytic results for query satisfaction and data routing in networks consisting of clients, sensors and data filtering/fusion servers. In this model, multiple clients pose queries that are satisfied by processing a set of sensor data streams through a set of filters or fuselets. Fuselets are lightweight data fusion algorithms that can be deployed in a network environment. The queries can have common sub-expressions (sub-queries) that should be reused by multiple clients if possible and appropriate. Moreover, effective routing of data streams from sensors to clients requires routing the streams through network nodes that can implement the required filtering/fusion operations. We formulate these problems quantitatively and propose a dynamic programming based solution using sensor-fuselet location and performance tables. The framework is preliminary in that many details and variations are abstracted or ignored. However, at the end of the paper we discuss several directions that can be explored to make these preliminary results more relevant to real scenarios.

1. Introduction

Several efforts are currently underway to design, build and/or deploy large-scale integrated sensor network, data fusion and command-control systems. Those efforts include:

- the Air Force's Joint Battlespace Infosphere (JBI) [1];
- DARPA's SensIT Program [2];
- DARPA's Information Exploitation Office (IXO) [3];
- NASA's JPL Sensor Webs Project [4].

Meanwhile several other research programs are organized around developing the basic technologies to support such systems. Among those research programs are:

- UC Berkeley's TINY Sensor project [5];
- DARPA Network Embedded Software Technology Program [6];
- Dartmouth's Sensor Web Project [7].

At Dartmouth, we have been working with about 100 re-configurable wireless sensor platforms that include: Global Positioning System (GPS) capabilities, wireless (RF) communications, iButton and serial sensor capabilities, simple microprocessor control (Intel 8051 processor) and self-contained power. One of the sensors is depicted in Figure 1 below. Details of these devices and the ad hoc routing algorithms for establishing wireless network connectivity can be found in [8].
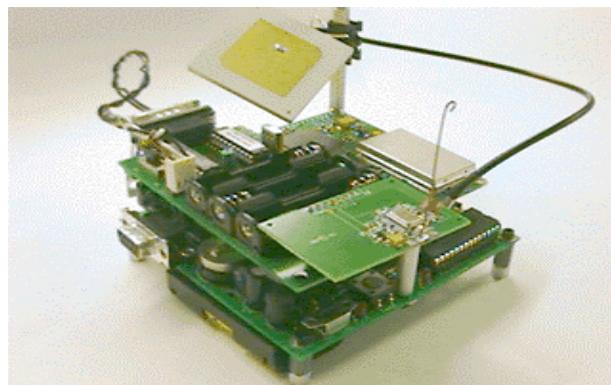

Figure 1: Dartmouth Re-configurable Wireless Sensor Platform

Deploying a complete, highly functional end-to-end system of such sensor nodes is a complex, multi-faceted problem. For example, sensors will likely be heterogeneous so some sort of self-defining registration protocol is required for maximal flexibility. Our early work is focused on a Sensor Markup Language, based on the DARPA Agent Markup Language (DAML) [9] for defining sensor capabilities in a standardized way. Another important technology, addressed in this paper, required to realize such system is to effectively and efficiently route sensor data streams to different clients through the network fabric. Ideally, those data streams would be processed, merged and multicast within the network to optimize some combination of performance metrics such as latency, bandwidth utilization or fuselet server loads.

In this paper, we first formulate a simple version of the sensor data stream routing problem, and then show how it

can be solved using dynamic programming ideas. Lastly we extend the basic model and framework to some more complicated situations.

2. Sensor Communication Networks

Our abstract model of a battlespace sensor communication network consists of the following ingredients: Clients, Fuselet Servers and Sensors. The abstract model of this network is illustrated in Figure 2, where C, N and S represent clients, fuselet server nodes and sensors, respectively.
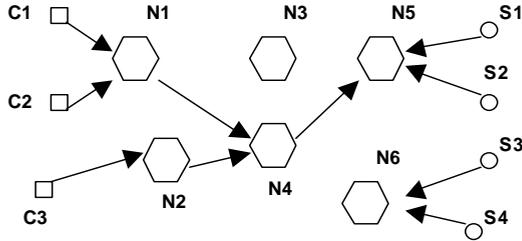

Figure 2: The abstract model of the network

The sensors in the battlespace include various sensing systems or devices such as radars, satellites, sonar, acoustic and other environmental monitoring devices. Such systems can generate a huge amount of radar, image, video, audio and other kinds of data from forward battlespace locations. Fuselets are programs that perform lightweight signal processing and data fusion operations on those streams. Network management operators are responsible for developing fuselets and uploading them to fuselet servers. When clients perform queries requiring battlespace information from the sensors, the sensor data streams are routed through the appropriate fuselet servers for data processing and then the processed data are returned to the clients.

In our framework, the fuselet server network is a peer-to-peer overlay network, which consists of geographically distributed fuselet server nodes. These server nodes are powerful computers with stable and high-speed network links between them. Subsets of these nodes may be managed by different organizations such as different services or agencies. The sensors describe themselves with the Sensor Markup Language (SML) using DAML and publish themselves to a local fuselet server. The SML-tagged description includes all the details about the sensor such as its capabilities, its functionality and its interfaces. The local fuselet server broadcasts the sensor information to all other fuselet servers. E.g. Sensor S1 and S2 publish themselves to their local access point – the fuselet server N5. Then N5 broadcasts their markup information to all other fuselet nodes. In the same way, when a network management operator develops a new fuselet, they use the Fuselet Markup Language (FML) to describe its capability, functionality and interface. While the fuselet

program is only uploaded to a subset of servers managed by these operators, the fuselet description information is broadcast to all other fuselet server nodes.

3. Sensor and Fuselet Tables

Every fuselet server in the network has a sensor information table that lists all sensors in this network, their SML tagged descriptions and their access points. That is, the server N1 has an abstract sensor table illustrated in Table 1.

```
S1:   SML-Description, N5
S2:   SML-Description, N5
S3:   SML-Description, N6
S4:   SML-Description, N6
         ......
```

Table 1: An abstract sensor table

Similarly, every fuselet server maintains a fuselet information table that lists all fuselets available in the network, their FML tagged descriptions and their host server network ID. An abstract fuselet table is shown in table 2.

```
F1: FML-Description
    : N1, N4, N6
F2: FML-Description
    : N2, N3, N4
       ......
```

Table 2: An abstract fuselet table

Since both the fuselet and sensor information are broadcast across the network, every fuselet node maintains the same sensor table and fuselet table at a generic operating time. However, for routing purpose, these tables may also include metrics to represent their routing preferences from every node's view, which differentiate these tables from each other. These sensor and fuselet tables are updated dynamically. Once a new sensor or fuselet goes online, the related information is automatically communicated and inserted into these tables. By contrast if a sensor or fuselet is not available anymore, the related information is automatically withdrawn from these tables.

4. Query Routing

For specific missions, clients submit queries for battlespace information to their local access points – local fuselets servers. Usually these queries are described in natural language or in high-level query languages. For example, client C1 may submit a query to node N1 and ask "Are there moving vehicles in battlespace region #1?". With all sensors and fuselets properly described in the corresponding markup languages, "composer" software we are developing will be able to discover specific sensor and

fuselet resources to satisfy the query. Based on semantic brokering, matching and reasoning, the composer assembles these sensors and fuselets into a sequence and creates a processing model. The processing model is interpreted and formatted into a Query Request Packet (QRP), which can be described in the following abstract format:

```
<Source, Fuselets-sequence>.
```

`Source` is the client's network identification, and `Fuselets-sequence` is the composed processing sequence of sensors and fuselets. The QRP may be decomposed into multiple sub-QRPs during the query routing process. For example, suppose that sensor S1 is an acoustic sensor and sensor S2 is a seismic sensor, which are both deployed in battlespace region #1 to detect moving targets. The server node N1 possibly decomposes C1's query into two sub-query request packets (sub-QRPs):

```
<C1,F1(F2(F8(S1)))>
```
and
```
<C1,F2(F5(F9(S2)))>.
```

The first QRP packet requests sensor S1 to route its data stream via fuselets F8, F2 and F1 for data processing before the processed data is returned to C1. The second QRP packet requests the sensor S2 to route its stream through fuselets F9, F5 and F2. Here we analyze a simple single-sensor filtering process first to illustrate the routing mechanism. Later in this paper, we will analyze the routing problem in the multi-sensor fusing process.

Since a fuselet is uploaded to several server nodes in the network for reliability and load balancing, the network must resolve the QRP's fuselets-sequence to physical server nodes that execute the fuselets. In this context, every node is not only a fuselet server but also a router. As in BGP routing (the Border Gateway Protocol (BGP) [10] is the main routing mechanism in Internet backbone routing), every server in the sensor network can choose one node from the node list as the best node to run each fuselet. In this case, the fuselet tables are not the same anymore since each node has a different view of the network. There are several routing mechanisms to compute a route, which are illustrated in Figure 3.

- *Forward Routing:* The client sends the QRP to the end sensor directly. The sensor sends its data stream to the network nodes capable of computing the first required fuselet processing. After one node invokes the first fuselet and processes the data stream, that node searches its fuselet table to find the best execution node for the next fuselet task listed in the QRP, and sends the processed data stream to that node. The new node continues this data processing and routing process. Eventually the processed data stream is forwarded to the requesting client.

- *Reverse Routing:* Just as in source routing mechanisms used in IP routing, the reverse routing approach resolves the route when the client forwards the QRP to the end sensors. At each hop, a node searches its fuselet table to find the best execution node for the next fuselet task, and sends the QRP to that node. The new node continues this QRP forwarding process and eventually the QRP is forwarded to the end sensor. The nodes in the route are sequentially added into the QRP. After the end sensor returns a data stream to the network, the data stream is routed through the nodes listed in the QRP in reverse order for data processing, and eventually the processed data stream is routed back to the client.
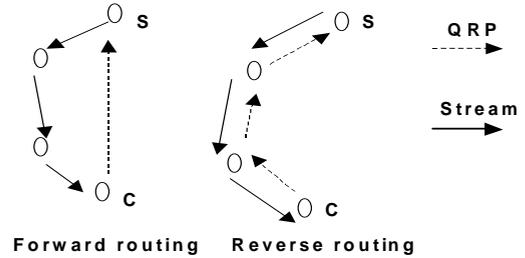

Figure 3: Different routing approaches

Compared to the IP routing, query routing in sensor networks has some significant differences. The data stream is not only routed but also processed sequentially along the route; since the sensor data stream usually has much larger amounts of data than the QRP, the data flow from sensors to clients is much bigger than the flow from clients to sensors. In sensor communication networks, we use reverse routing to improve the network performance. A client forwards the QRP via fuselet nodes to the end sensor in reverse routing. Every node in the route maintains a QRP waiting table to list its pending QRPs. These fuselet nodes are aware of the required fuselet computing tasks before the sensor data streams are fed to them. That is, the client must "make a reservation" on those nodes before the nodes can offer the specific fuselet computing services. Those nodes can invoke resource control to receive and process the large amounts of sensor stream data passing through them.

The real advantage of reverse routing as described here is to merge overlapped QRPs in the network, which is illustrated in Figure 4. In Figure 4, the client C1 has a sub-query QRP `<C1,F1(F2(F8(S1)))>` resolved by the network as `<C1,F1@N1(F2@N4(F8@N5(S1)))>` and the client C3 has a sub-query QRP `<C3,F9(F2(F8(S1)))>` resolved by the network as `<C3,F9@N2(F2@N4(F8@N5(S1)))>`, where Fx@Ny means that the fuselet Fx is assigned to execute at the fuselet node y. After client C1 and server N1 forward C1's sub-QRP `<N1,F2(F8(S1))>` to N4, at first N4 adds `F2<-F8<-S1` to its QRP waiting table and then forwards the sub-QRP `<N2,F8(S1)>` to N5. N5 continues this process. Meanwhile N2 forwards C3's sub-QRP

`<N2,F2(F8(S1))>` to N4. When N4 tries to insert the new sub-QRP `F2<-F8<-S1` to its QRP waiting table, it finds that it has already been routing and processing this data stream from sensor S1. N4 will not forward C3's sub-QRP to F8 and S1 anymore. Instead it just adds N2 into its source address and later multicasts the processed data stream to both N1 and N2. The overlapped QRPs are merged in the middle of the route and the data stream will not be pulled and processed twice along that route. Sensors devices can output large amount of data using much slower links in the network. This QRP merging process can dramatically reduce both network traffic and computation loads on the fuselet nodes.
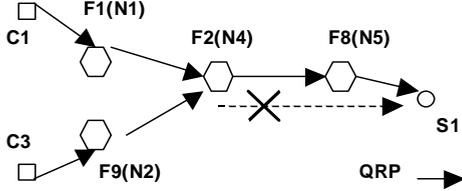

Figure 4: Merge the overlapped QRPs

5. Routing Optimization

Based on a local preference, every node can choose one node as the best node or default node to run each fuselet from its view of the network. If a node is not available, another node takes its place. Although this approach is very reliable in dynamic networks, this routing algorithm is similar to a greedy search algorithm. The sensor query routing optimization we describe is more complicated than existing IP routing optimization such as distance vector routing. The sensor data stream is not only routed but also processed along the route. Query latency is due to both data communication and fuselet computation along the route. Since a query resolves to a specific fuselet sequence, each query might have to form a specific routing path even though they can have the same source and destination nodes. In this section, we discuss how to minimize query latency under some assumptions.

Here Nc and Ns are used to represent the network access points of client C and sensor S respectively. Nc submits a QRP:

$$<C,Fx_1(...Fx_i(...Fx_m(S)...)...)>$$

for client C, where Fx_i represents fuselet x_i and $1 \le i \le m$. Now we use $S(Fx_i)$ to represent the subnet of all nodes that have the fuselet Fx_i, and use n_i to represent the node number in this subset $S(Fx_i)$. For every QRP, the routing process is to find the best path from node Ns to Nc via these subsets, which is illustrated in Figure 5. The total search space for the routing path is $\prod_{i=1}^{m} n_i$.

In order to minimize query latency, we will make the following assumptions:

Assumption 1: Every fuselet node maintains an up-to-date table that lists the current communication latency between every pair of fuselet nodes in the network.
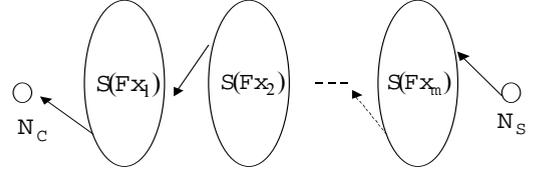

Figure 5: the routing optimization process

Assumption 2: Every node's fuselet table maintains the up-to-date fuselet computation time on their hosted nodes.

Theorem 1: Under Assumptions 1 and 2, for any QRP $<C,Fx_1(...Fx_i(...Fx_m(S)...)...)>$, the path with minimal query latency can be found for reverse routing with computational complexity $o(\sum_{i=0}^{m} n_i n_{i+1})$.

Proof: For any query request packet – QRP $<C,Fx_1(...Fx_i(...Fx_m(S)...)...)>$, the access point Nc of client C can search its sensor table and find sensor S's access point Ns. With its fuselet table, Nc can form a directed multipartite graph G(V, E) from node Ns to Nc as illustrated in Figure 5. The graph includes m layers of nodes between Nc and Ns, with n_i nodes in layer i. Here we define Nc as layer 0 with node number $n_0 = 1$ and define Ns as layer m+1 with node number $n_{m+1} = 1$. $V_{i,j}$ is used to represent node j at layer i, where $0 \le i \le m+1$ and $1 \le j \le n_i$. Define the communication time between nodes $V_{i,j}$ and $V_{i-1,k}$ as $C(V_{i,j},V_{i-1,k})$, where $1 \le j \le n_i$, $1 \le k \le n_{i-1}$ and $1 \le i \le m+1$. The communication time is the time used for two nodes to exchange the data stream and QRP packet, which can be computed based on Assumption 1 and the data stream size. If $V_{i,j}$ and $V_{i-1,k}$ are the same physical server, that is two fuselets running on the same machine, we define their communication time as zero. We define the processing time of fuselet Fx_i on the node $V_{i,j}$ as $P(V_{i,j})$, which can be computed based on Assumption 2. Since there is no computation on the Nc node in the layer 0, we define $P(V_{0,j}) = 0$. Edge weights for the graph G(V,E) can be defined as:

$$\forall 1 \le j,k \le n_i, 0 \le i \le m+1,$$
$$d(V_{i,j},V_{i,k}) = +\infty;$$
$$\forall 1 \le j \le n_i, 1 \le q \le n_p, 0 \le i,p \le m+1, \text{ if } p \ne i-1,$$

$$d(V_{i,j}, V_{p,q}) = +\infty \ ;$$
$$\forall 1 \le i \le m+1, 1 \le j \le n_i, 1 \le k \le n_{i-1},$$
$$d(V_{i,j}, V_{i-1,k}) = C(V_{i,j}, V_{i-1,k}) + P(V_{i-1,k}) \ ,$$
$$P(V_{0,j}) = 0 \ ;$$

We have thus formulated the query latency optimization problem as a classic shortest path problem. The path with the shortest distance from node $V_{m+1,1}$ to node $V_{0,1}$ in the graph is the path with the minimal query latency in the network. To solve this shortest path problem, we can use, for example, the Bellman-Ford algorithm [11] and reduce the computing complexity from $o(\prod_{i=1}^{m} n_i)$ to $o(\sum_{i=0}^{m} n_i n_{i+1})$.   ∎

In order to maintain network connectivity, each fuselet server periodically sends "probing" or "keep-alive" messages to its peers. The latency between two peers can be measured with these existing "probing" messages. To satisfy Assumption 1, each node has to broadcast its communication latency to other peers in the network, which causes extra-overhead in network communication. However, if there are only dozens of fuselet nodes in the network, the broadcast will add only a small amount of network traffic due to this overhead. Assumption 2 can be satisfied if each node broadcasts its fuselet execution time periodically within the peer-to-peer network. This broadcast process also adds communication overhead to the network.

Although Theorem 1 theoretically solves the optimal routing problem, this routing mechanism is not practical in a real sensor network. It is difficult to measure and broadcast fuselet computing times and peer-to-peer communication times so dynamically. The challenge in query latency optimization is that we have to use the same metric - time, to measure the latency due to fuselet computation and network communication. In the distance vector routing approach, "hops" are used to count the distance between two nodes. In the task scheduling, "CPU Usage" and "Memory Usage" are used to represent the task load on a machine. Since query latency arises from both task computation and data communication, it is not clear how to combine metrics, namely "hops" and "CPU usage", to represent the latency. However, if we are not seeking the optimal path but a "good" path in query routing, we can use some static metrics to represent those dynamic parameters [12]. For example, the bandwidth between two peers can be used to represent the communication latency between them; a server node's CPU speed and memory size can be used to represent the fuselet computing time on that node.

6. Dynamic Multi-Sensor Fusion

We have analyzed the simple query routing scenario in sensor communication network: A single sensor's data stream goes through fuselets sequentially for data processing and eventually the processed data is returned to the client. In fact a fuselet is more like a filter in this situation. The query routing process is more complicated in a multi-sensor fusion process. However, our routing scheme and optimization results can be readily extended to such more complicated situations.

- *A sensor has multiple access points.* A sensor can have multiple access points to the network. For a QRP routed to this sensor, in the routing decision process, we just need to add another layer of nodes between the sensor and the last layer $S(Fx_m)$. Theorem 1 applies to this situation if the communication latencies from the sensor to these access points are known.

- *A fuselet integrates multi-sensor's data stream.* If a fuselet only integrates and fuses data streams for several specific fixed sensors, we can consider this fuselet to be an integrated sensor instead of a dynamic fuselet program. This fuselet publishes itself as a new sensor in the network.

- *Dynamic multi-sensor fusion.* In a dynamic multi-sensor fusion process, the client's queries are represented as more complicated QRPs such as QRP0:

```
<C,F2(F4(F5(F6(S1)),S2),S3)>
```

QRP0 requests sensor S1's data to be processed via the fuselets F6 and F5 sequentially. That processing result and sensor S2's data stream are fed to the fuselet F4. After that, F4's computing result and sensor S3's data stream are fed into fuselet F2 and the process continues. In the reverse routing illustrated in Figure 6, at first the node Na forwards the QRP0 to node Nb with fuselet F2. Then Nb splits the QRP0 to two sub-QRPs: QRP1 `<Nb,S3>` to the sensor S3 and QRP2 `<Nb,F4(F5(F6(S1)),S3)>` to a node Nc with fuselet F4. Nc splits the QRP2 to another two sub-QRPs: QRP3 `<Nc,S2>` to the sensor S2 and QRP4 `<Nc,F5(F6(S1))>` to a node Nd with fuselet F5. The routing process continues to forward the sub-QRP until it gets to the end sensor S1. Based on this approach, some QRPs are actually processed concurrently such as QRP3 and QRP4.

If every node chooses a best node or default node to run each fuselet based on a local preference, the routing process in multi-sensor fusion is the same as that in single sensor fusion. For the optimal routing process, Theorem 1 can still be applied in multi-sensor fusion although we need to change the latency computation method for parallel QRPs.
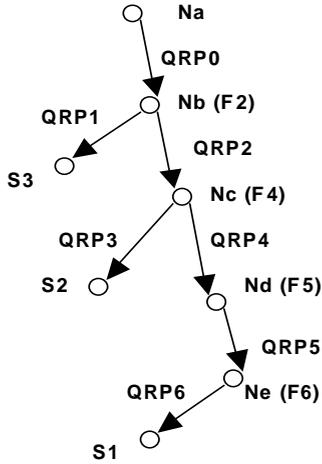
Figure 6: An example of QRP routing
in multi-sensor fusion

Here we use $L(QRP_i)$ to represent the minimal query latency for the query request packet QRP_i. At fuselet node Na illustrated in Figure 6, the minimal query latency $L(QRP_0)$ can be computed from with the following relationship:

$$L(QRP_0)$$
$$= \min_{N_x} (\max(L(QRP_1), L(QRP_2))$$
$$+ P(F2 @ Nx) + C(Na, Nx))$$

where $P(F2 @ Nx)$ is fuselet F2's computing time at node Nx and $C(Na, Nx)$ is the data communication time between two nodes Na and Nx. Furthermore, we can compute $L(QRP_1)$ and $L(QRP_2)$ from similar equations. With these recursive equations, it is clear that the Bellman-Ford algorithm can be applied to solve this optimal routing problem. However, we should expect that multiple optimal paths may exist in this routing process because of the "max" operation in the above equations.

7. Conclusions

In this paper, we have presented a framework and some analytic results for query satisfaction and data routing in sensor communication networks. In our model, sensor data streams are routed through specific fuselet sequences for data processing before the processed data are returned to the clients. With our routing mechanism, multiple clients' overlapped sub-queries can be merged along the route to increase network utilization and performance. Furthermore, we have described a dynamic programming based solution for the optimal routing problems in multi-sensor fusion.

References

[1] Air Force Joint Battlespace Infosphere (JBI) homepage: http://www.rl.af.mil/programs/jbi/default.cfm

[2] DARPA SensIT Program homepage: http://dtsn.darpa.mil/ixo/sensit%2Easp

[3] DARPA Information Exploitation Office (IXO): http://dtsn.darpa.mil/ixo/

[4] NASA-JPL Sensor Webs Project homepage: http://sensorwebs.jpl.nasa.gov/

[5] UC Berkeley Sensor Project: http://today.cs.berkeley.edu/800demo/

[6] DARPA Network Embedded Software Technology Program: http://www.darpa.mil/ito/research/nest/index.html

[7] Dartmouth's Sensor Web Project: http://www.ists.dartmouth.edu/IRIA/projects/sensor_web.htm

[8] Michael Corr, Masters Thesis, Thayer School of Engineering, Dartmouth College, Hanover, NH 2001 http://actcomm.dartmouth.edu/~mgcorr/papers/Corr_Thesis.pdf

[9] DARPA Agent Markup Language (DAML) Program: http://www.daml.org

[10] S. Hablabi and D. McPherson, *Internet Routing Architectures*, second edition, Cisco Press, 2000.

[11] D. Bertsekas and R. Gallager, *Data Networks,* second edition, Prentice Hall Inc., 1992.

[12] David Kotz, George Cybenko, Robert Gray, Guofei Jiang, Ronald Peterson, Martin Hofmann, Daria Chacon, Kenneth Whitehead and Jim Hendler, *Performance Analysis of Mobile Agents for Filtering Data Streams on Wireless Networks*, ACM Mobile Networks and Applications Journal, vol.7, no.2, March, 2002.