Floating car data for traffic monitoring

Torp, Kristian; Lahrmann, Harry Spaabæk

Published in:
ITS at the Crossroads of European Transport

Publication date:
2005

Document Version
Publisher's PDF, also known as Version of record

FLOATING CAR DATA FOR TRAFFIC MONITORING

Kristian Torp
Department of Computer Science
Fredrik Bajers Vej 7E
DK-9220 Aalborg Ø
Aalborg University, Denmark
torp@cs.aau.dk
www.cs.aau.dk/~torp

Harry Lahrmann
Traffic Research Group
Fibigerstræde 11-13
DK-9220 Aalborg Ø
Aalborg University, Denmark
lahrmann@plan.aau.dk
www.plan.aau.dk/~lahrmann

ABSTRACT

This paper describes a complete prototype system that uses Floating Car Data (FCD) for both automatic and manual detection of queues in traffic. The system is developed under EU's Tempo program. The systems consists of small hardware units placed in mobile traffic report units (we use taxis) and backstage databases that collect all data send from the report units. The communication between the taxis and the databases is based on a very compact wireless communication protocol. A one month field test using ten taxis shows that the system is operational and that communication costs are very low (33 euro per taxi per year).

PROJECT BACKGROUND

Sensors, such as loops, embedded in a road network have shown to be able to provide precise and timely information on the current traffic situation that for example can be broadcasted via public service and private radio stations. Of the highest interest is reporting of queues in traffic, i.e., traffic jams. The main problem with sensors is that they are expensive to repair and maintain. Of this reason it is very interesting to look into alternatives.

The purpose of the REMOTE project is to look at using FCD as an alternative to road sensors for traffic monitoring. In particular, the purpose is to use FCD for both automatic and manual detection of traffic queues. The automatic detection is based on analyzing GPS data from mobile traffic report units (we use taxis in the project therefore these units are called taxis in the following). The manual detection is based on taxi drivers reporting traffic queues by using new equipment installed in the taxis. This equipment has been developed as part of the REMOTE project.

The members of the projects are two private companies and two research groups from Aalborg University. The company M-tec has developed the equipment installed in the taxis and the company Euman has handled the storing and retrieval of the FCD data in the backstage databases. The two research groups involved are the Traffic Research Group and the Department of Computer Science's Database Group.

SYSTEM ARCHTECTURE

The overall architecture of the system used in the REMOTE project is shown in Figure 1.

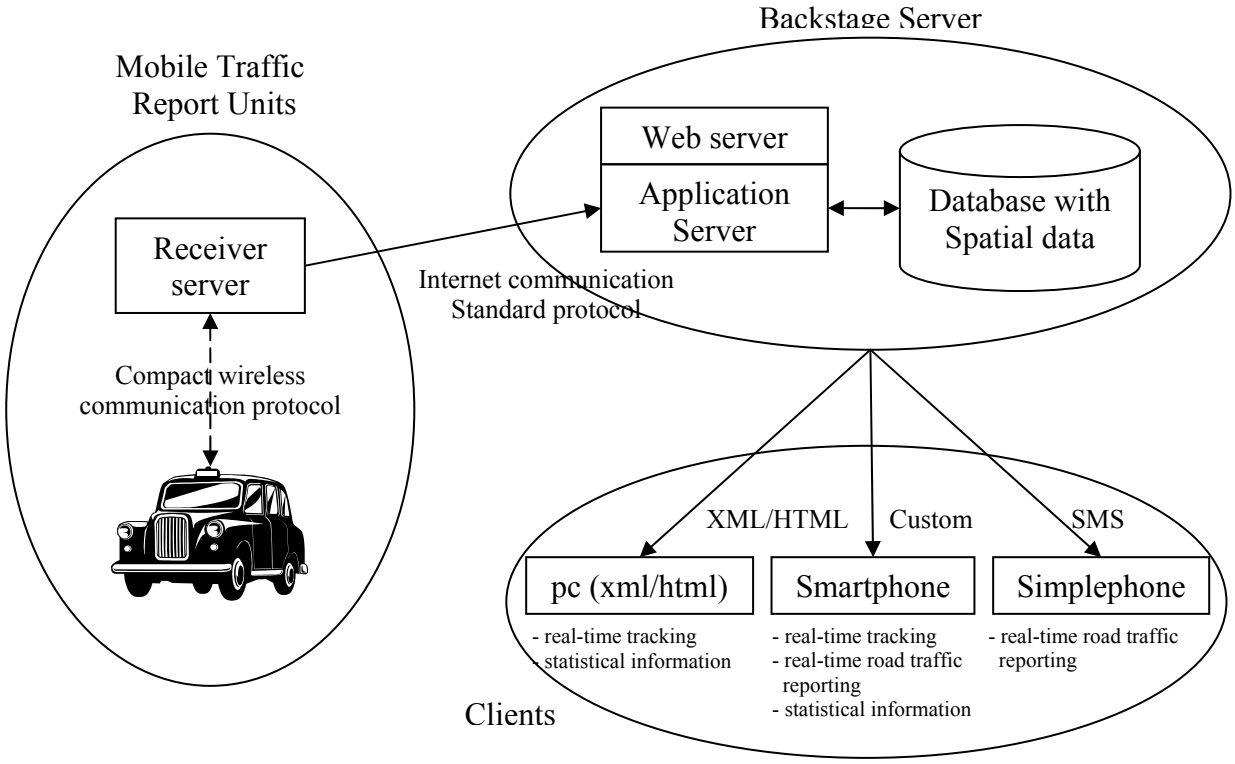

Figure 1 The overall system architecture in the REMOTE project

To the left the taxis with special hardware are shown. Note that the taxis communicated with a dedicated receiver server via a special compact wireless communication protocol. Every 5 seconds all taxis sends a small message to the receiver server reporting on each taxis position, speed and so on. In addition, the taxi drivers can manually report traffic queues using the small display shown in Figure 2. When a taxi enters a traffic queue the driver presses the right (red) button. When the taxi is no longer in a traffic queue the driver presses the left (green) button.

Figure 2 The display for manual queue reporting

The receiver server communicates with a backstage server (upper right corner of Figure 1) that collects and stores all data from the taxis. The communication between the receiver server and the backstage server is based on the standard HTTP protocol and uses the standard Universal Resource Indicators (URIs) message format. A URI message contains the same information as the compact messages send between the taxi and the receiver server but is 10-20 times larger. The backstage server stores and indexes all the data received from the taxi using Oracle database technology this includes extensions to handle spatial data.

To report traffic queues and related information to clients currently driving or are planning to drive the backstage server can deliver information to three different types of clients that are shown in the bottom right of Figure 1. The first type of client is a pc that can receive real-time tracking information on taxis but also statistical information, e.g., time, date, and number of traffic queues.

These clients are assumed to have large displays. The second type of client is the smartphone that can receive basically the same information as the pc clients but the protocol used to distribute the information can be tailored, e.g., to a specific smartphone platform such as the Nokia's Series60. To be able to communicate with the broadest range of clients the last type of client supported is a simplephone. These clients can only receive simple messages with real-time road traffic reports such as "Queue on Highway E45 North Direction between intersection 26 and 27". These messages are sending as SMS messages.

TRAFFIC QUEUE DETECTION ALGORITHM

To be able to detect traffic queues we define the following two concepts. *Measuring stretch* which is a part of the road network where it is assumed that cars will always drive the maximum allowed speed unless there is traffic queues. These measuring stretches are selected such that they are not close to road intersections, parking lots and other obstacles that may slowdown traffic. *Report stretch* which is a part of the road network well-know to the road users. This can for example be a road stretch between to major intersections. Note that a there can be more than one measuring stretch for a report stretch. Figure 3 shows an example of a road network where roads are shown as lines (both solid and dashed lines). The measuring stretches are shown as rectangles and report stretches are indicated by brackets. One of the report stretches is between a major road intersection and a roundabout. This report stretch contains two measuring stretches.

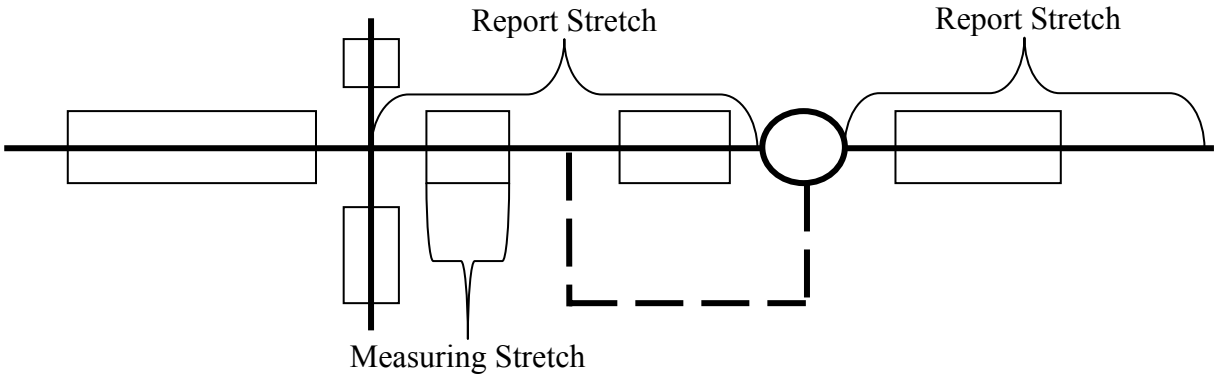

Figure 3 Road Network with Report and Meassuing Stretches

The central idea in the traffic queue detection algorithm is to detect traffic queues on a report stretch if there is a queue on any of the measuring stretches on the report stretch. The main challenges in designing such an algorithm are the following. (1) how to make the algorithm scale to a large road network, (2) how to use both manual queue detections (i.e., when the queue button in Figure 2 is pressed) and FCD such as GPS positions and current speed of the cars, and (3) how to report in a timely manner when queues are created and dissolved. These challenges will be address in the following subsections.

HANDLING LARGE ROAD NETWORKS

To be able to make the algorithm scale to large road networks we choose to only look at the major roads in the network. As shown in Figure 3, the road indicated by a dashed line is not a report stretch as is does not have any measuring stretches. The roads that are excluded are typical the

minor roads with little traffic, e.g., in suburbs or rural areas. The major roads picked to be used are selected based on existing knowledge on the road network, e.g., which road are already known to have, or likely to have, traffic queues. In this way it is possible to filter the GPS positions. Only the GPS positions that are map to be within a measuring stretch are used in the queue detection algorithm all other GPS positions are discarded. In the field test 28% of the GPS positions were within a measuring stretch.

COMBINING MANUAL AND AUTOMATIC DATA

To detect traffic queue we have data provided manually by the taxi drivers (pressing the queue button) and we have automatic data provided as GPS data. To get the best quality in reporting traffic queues these two data sources need to be combined. We assume that the manual data is a more reliable source for traffic queue detection than the GPS data. For this reason we assign different weights to these two data sources. The weights used are shown in Table 1. As can be seen we put the highest weights on the user reported queue started and dissolved, the values 10 and -10, respectively. The remaining weights are according to [1] where no congestion gets the weight -2 and critical congestion gets the weight 4. How these weights are used in explained in the next section.

Congestion Level	Weight
User reported queue dissolved	-10
No congestion	-2
Insignificant congestion	0
Initial congestion	1
Huge congesion	2
Critical congestion	4
User reported queue started	10

Table 1 The Congestion Levels and Their Weight

REPORTING TRAFFIC QUEUES

It is important that the congestion level calculated by the queue detection algorithm corresponds with the congestion level in real-life. As an example, if taxi stops and picks up a customer in a measuring stretch where the taxi is not allowed to stop the queue detection algorithm must not immediately report critical congestion. To avoid such situation a time delay is build into the algorithm.

The time delay is build into the algorithm by considering both the most recent data received and older data. This is illustrated in Figure 4 where a data queue is used to store the last eight congestion

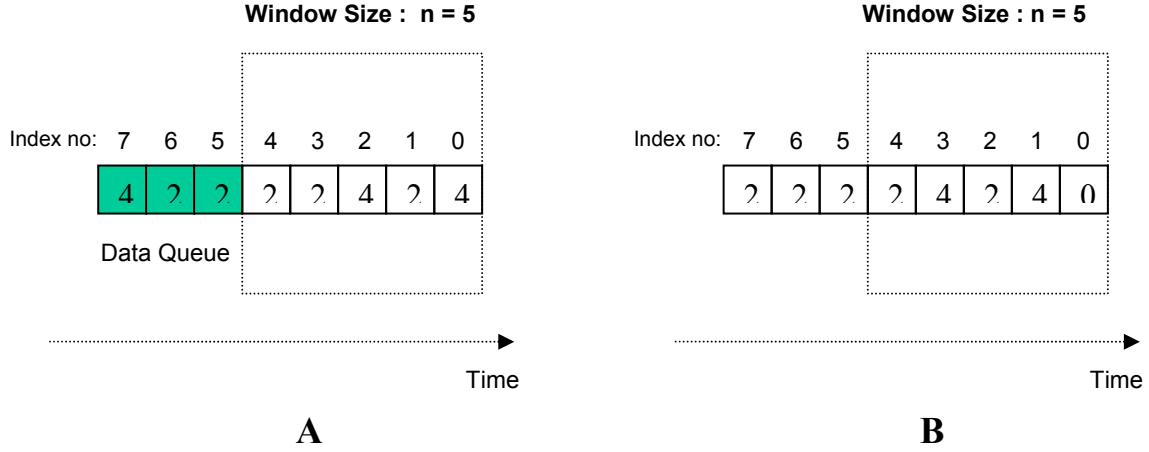

Figure 4 Data Structures for Computing General Congestion Levels

levels. In the following, we assume that the data queue stores a single congestion level value for each minute. Currently the congestion level is critical, i.e., the value 4 in the data queue entry with index number 0. A minute ago the congestion level was huge, i.e., the value 2 in the data queue entry with index number 1, and so on. Note that the congestion level stored in the data queue is an aggregation of all the data report within the minute that the entry represents, recall that new GPS data is received every five seconds.

To detect traffic queue within a measuring stretch each of these are associated with a data queue as shown in Figure 4. Naturally, the GPS and manual data only effects the congestion level of the measuring stretch to which the data is map match.

The current aggregated congestion level for the last minute is to fine-grained for detecting traffic queues. We need to consider more values. Figure 4A shows a window that spans the five most recent congestion levels computed. To build a time delay into the congestion level reported, called the *congestion value* this value calculated on the basis of these last five computed congestion levels in the data queue. This calculation is a weight average where the most recent computed congestion levels are weighted higher than the older congestion levels. The weight used in shown in Table 2. The current congestion level has the weight 0.5 in the computation of the congestion value, the congestion level a minute ago has the weight 0.2, and so on. Note that the sum of the weights should be 1.0.

Index no.	Weight
0	0.5
1	0.2
2	0.1
3	0.1
4	0.1

Table 2 Index Weights

The congestion value in Figure 4A is calculated in Table 3. The congestion value is 3.2 which is between a huge and a critical congestion level, see Table 1.

Index no.	Value	Weight	Value * Weight
0	4	0.5	2.0
1	2	0.2	0.4
2	4	0.1	0.4
3	2	0.1	0.2
4	2	0.1	0.2
		Congestion value	**3.2**

Table 3 First Congestion Value Computation

As time passes new congestions levels are calculated and inserted in the front of the data queue (index number 0) in Figure 4. The remaining values are shifted one position to the right. Figure 4B shows the congestion levels a minute later than Figure 4A does. The most recent congestion level is insignificant congestion (the value 0). The new congestion value is calculated in Table 4. This value is between initial and huge congestion.

Index no.	Value	Weight	Value * Weight
0	0	0.5	0.0
1	4	0.2	0.8
2	2	0.1	0.2
3	4	0.1	0.4
4	2	0.1	0.2
		Congestion value	**1.6**

Table 4 Second Congestion Value Computation

CONFIGURATION OF ALGORITHM

The queue detection algorithm can be configured in numerous ways to adapt to real life. Most notably the congestion level weights in Table 1 can be changed, the window size in Figure 4 can be made shorter or longer, and the index weights in Table 2 can be changed.

FIELD TEST

A one month field test of all the components in Figure 1 was conducted in March 2004 using ten taxis in the Aalborg city region. The taxi drivers were briefly instructed on how to use the system. In particular, the drivers where instructed on how and when to use the manual queue reporting display shown in Figure 2. Because not all roads are interesting with respect to traffic queue detection only the main roads in the city of Aalborg are used. The actual road network used is shown in Figure 5. A measuring stretch is indicated by the bold coloured lines. A report strecth is indicated by a number. Note that there can be several measuring stretches on a report stretch. In Figure 5, the city limits of Aalborg and its suburbs are shown using the light-green color. The light-blue color shows the fjord called Limfjorden that separates the northern part of the Aalborg from the southern part.

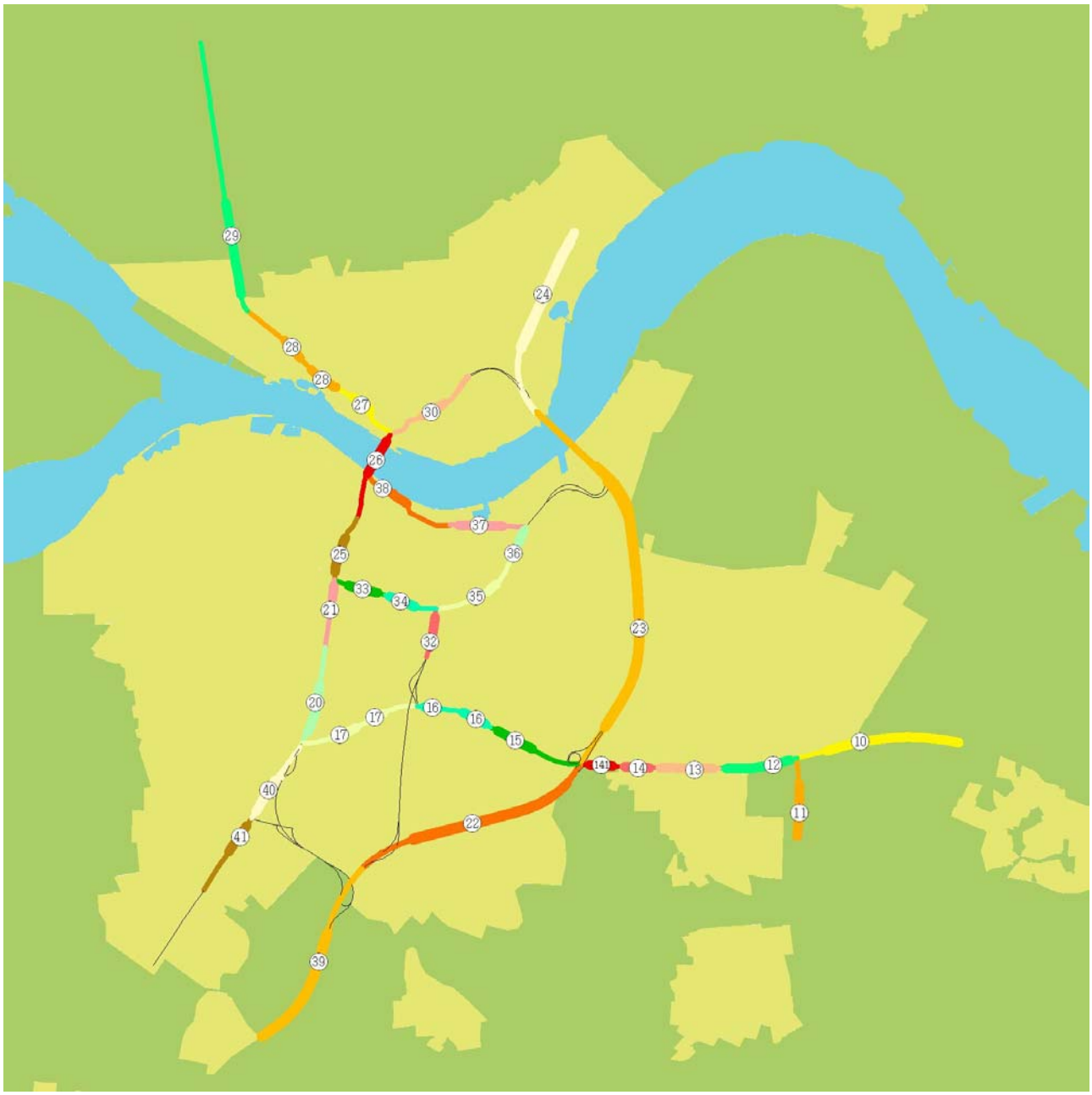

Figure 5 The Road Network, Measuring, and Report Stretches

RESULTS

For the entire field test the system worked without hardware or software failures. During the test 465,000 messages where send that formed the basic for the automatic traffic queue detection. The taxis send between 4,787 and 66,489 messages each. Further, the taxi drivers reported manually 176 queues entered but only reported manually 151 queues exited.

After the field test was finished all the messages were mapped to a digital road network using existing map match algorithms[2]. Of the 465,000 messages 131,501 messages were on the road network shown in Figure 5.

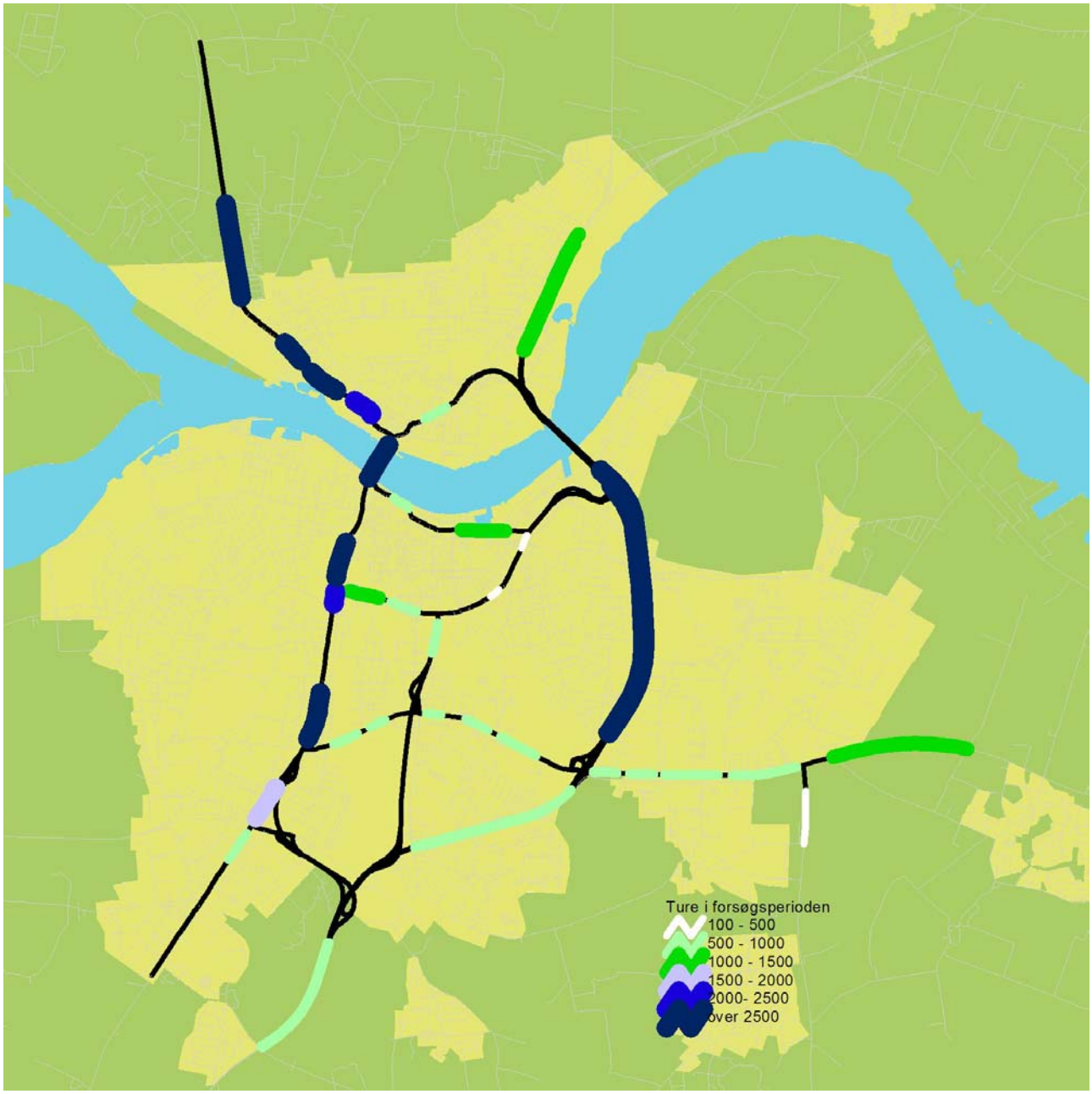

Figure 6 Messages Mapped to Road Network

In Figure 6 the total number of messages is mapped to the road network used. The dark blue color indicates more traffic. In general the north-south going roads are best covered (report stretches 20, 21, 23, 25, 26, 28, and 29 in Figure 5). These are the roads going toward the only two ways of passing the fjord separating the top and bottom half of the map. The east-west going roads (report stretches 12 to 17) are not covered as well as the north-south going.

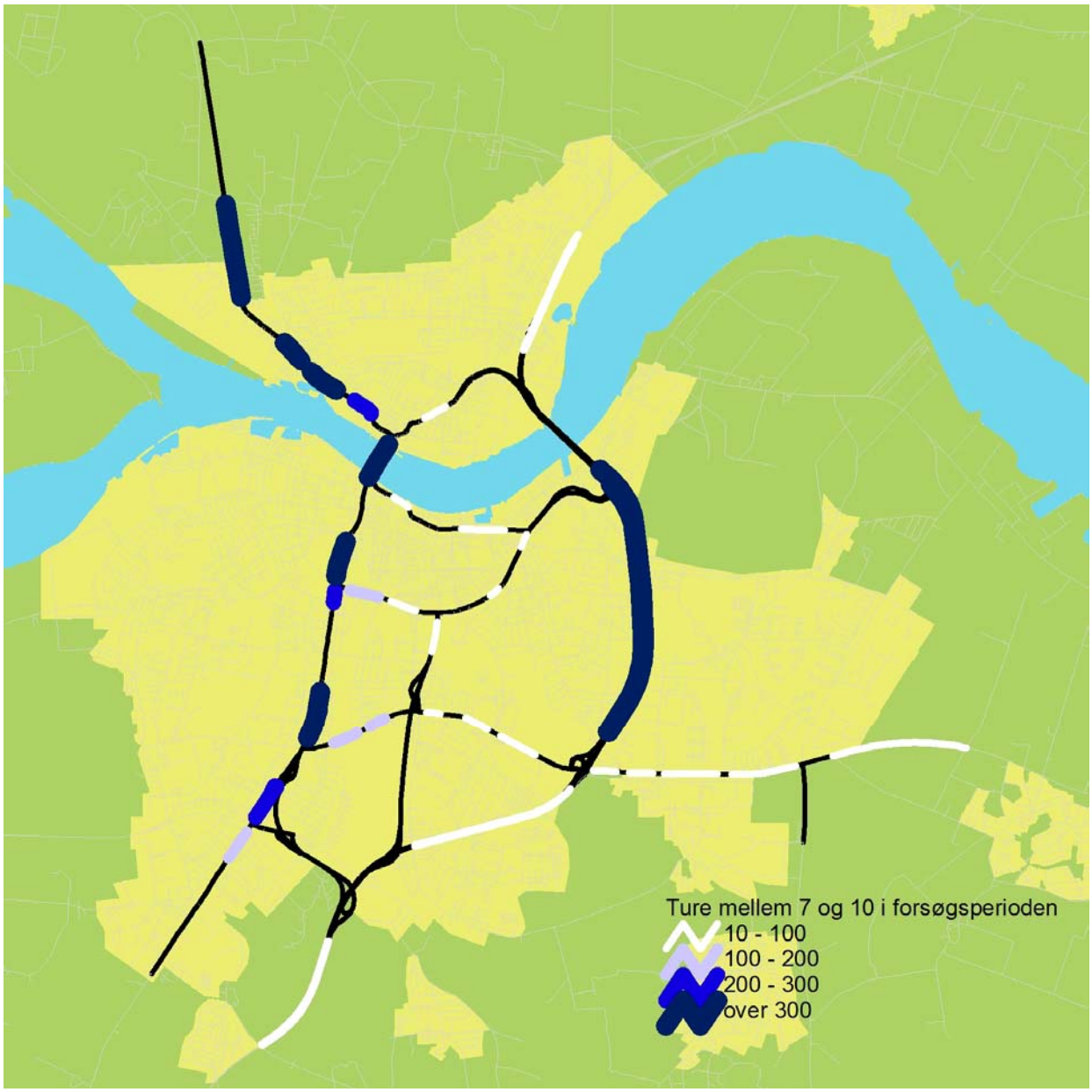

Figure 7 Morning Peekhour Traffic on Weekdays between 7.00 – 10.00

Figure 7 shows the morning peekhour traffic. This traffic situation is consistent with the total number of messages shown in Figure 6. Again the north-south going roads are better covered than the east-west going roads. Here the dark blue color also indicates more traffic.

The afternoon peekhour traffic is shown in Figure 8. Like the morning traffic shown in Figure 7 the north-south roads have the most traffic. However, in the afternoon the east-west going traffic is higher than in the morning. This is unexpected.

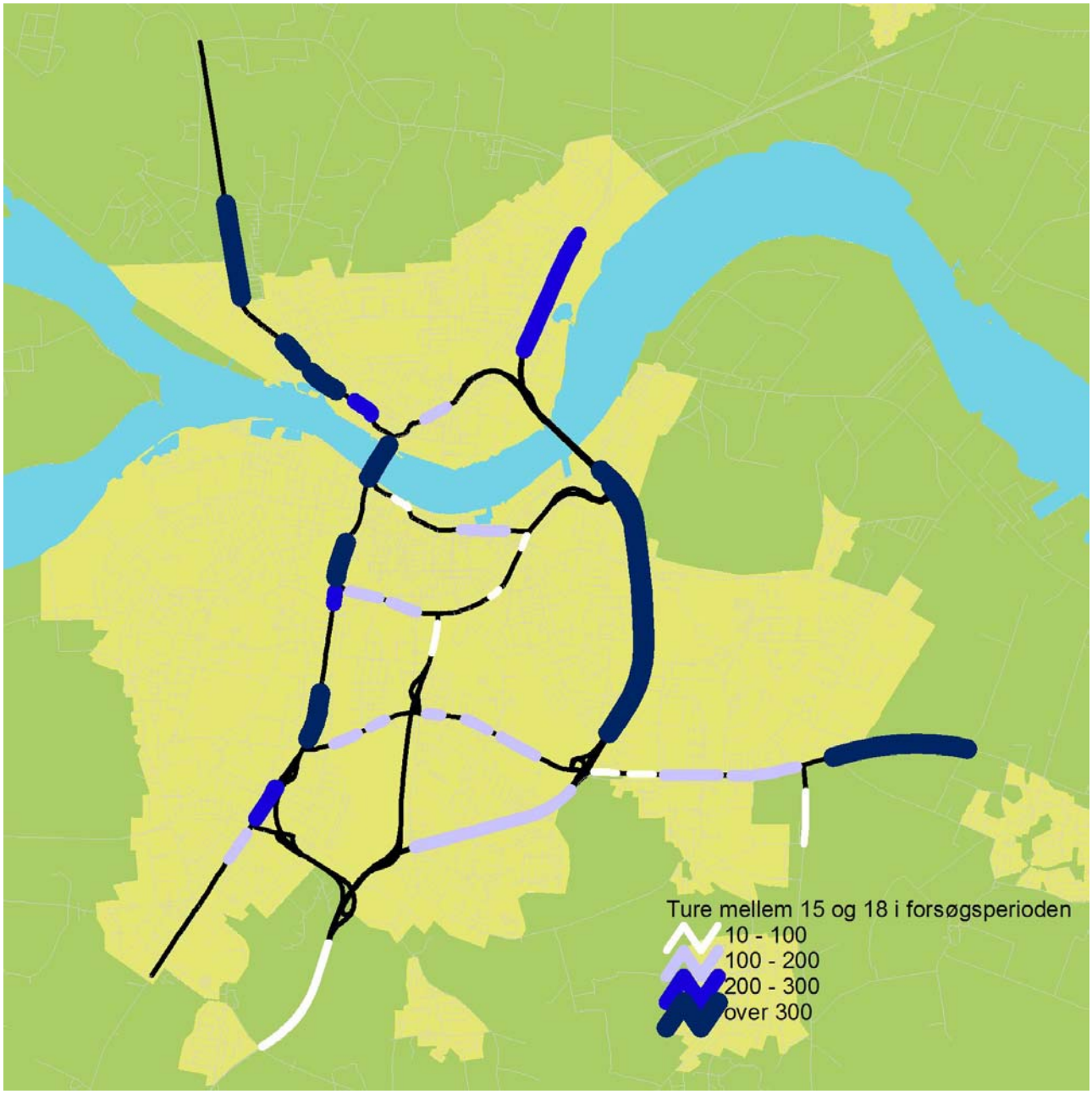

Figure 8 Afternoon Peekhour Traffic on Weekdays 15.00 – 18.00

Overall for the field test, was it quickly discovered that ten taxis are too few to detect traffic queues in real-time when the cars drive in an area as big as it was the case for this field test. Therefore we have not been able to test the traffic queue detection algorithm using real data. However, the automatically collected data can be used to generate statistical visual report that shows when and were traffic queue occur, as shown in Figure 7 and Figure 8. The manual queue detection worked, however, it is a problem that in some cases the drivers reports a queue but forgets to report that the queue dissolved.

In addition to this, the field test showed that it is possible to operator a single taxi that sends messages every 5 seconds for only 33 euro/year. This is possible due to the very compact communication protocol used between the taxis and the receiver server in Figure 1.

CONCLUSION

Floating car data (FCD) is an interesting new wireless technology that can be used to report traffic queues. In this paper, we have shortly described a complete prototype system that demonstrates that FCD can be used for traffic queue detection. A field test has shown that the system is functional and very cheap to use in terms of communication costs. However, the field test also generated a number of new questions. In particular, how many taxi are need to do real-time traffic queue detection, how to combine automatic and manual queue detection, and how to integrate the FCD data with existing traffic queue detection systems such as sensor embedded in the roads.

ACKNOWLEDGEMENT

The project is cofunded by the European Commission, DG Tren, TEMPO PROGRAMME
MIP 2003 VIKING A EURO-REGIONAL DEPLOYMENT PLAN FOR ROAD ITS BETWEEN NORTHERN GERMANY, DENMARK, FINLAND, SWEDEN AND NORWAY.

REFERENCS

[1] Highway capacity manual. - Washington, D.C. : Transportation Research Board, National Research Council, 2000-. - 1 v. (loose-leaf) : ill., 30 cm. National Research Council (U.S.). Transportation Research Board

[2] Jens, Juhl, INFATI – Mapmatching, Notat 3, Institut for Samfundsudvikling og Planlægning, Aalborg Universitet, 2001.