UNIVERSITY OF CINCINNATI

Date: _____

I, _____,
hereby submit this work as part of the requirements for the degree of:

in:

It is entitled:

This work and its defense approved by:

Chair: _____

Performance Evaluation of Real-time Event Detection Algorithms

A thesis submitted to the Graduate School

of the University of Cincinnati

In partial fulfillment of the requirements for the degree of

Master of Science

In the Department of Computer Science of the College of Engineering

August 2006

By

Katherine A. Umberg

Bachelor of Science, Mathematics

Xavier University, Cincinnati, OH

May 2002

Committee Chair: Professor Carla Purdy

Abstract

Recent terrorist attacks have focused attention on the safety of water distribution systems. Since it is implausible to protect these networks from contamination, a Contamination Warning System (CWS) is needed to detect contamination so that attacks can be contained. A CWS consists of several interrelated components, including a network of sensors, methods for collecting and analyzing their data, and response procedures for abnormal conditions.

Effective sensor data analysis is essential for successful CWS's: even perfect sensors in ideal locations are useless unless their data is effectively processed and utilized. This requires analysis of large amounts of imperfect water quality sensor data to distinguish between normal water quality variability and that due to a foreign substance.

This work proposes an objective, realistic experimental testbed for evaluating contamination detection algorithms. It can begin a recursive process of algorithm testing and refinement, culminating when a sufficiently accurate and reliable algorithm is developed.

Contents

1	Introduction	9
2	Objectives	13
2.1	Motivation	13
2.1.1	Contamination Warning Systems	13
2.1.2	Data Analysis	16
2.2	Project Objectives	20
2.3	Research Significance	22
3	Related Work	23
3.1	Event Detection Algorithms	23
3.2	Previous Detection Algorithm Evaluations	26
3.2.1	Bio-ALIRT	28
3.2.2	DARPA Offline Intrusion Detection Evaluations	34
3.3	Proposed Evaluation Frameworks in Other Fields	39
3.3.1	CDC Recommendations	40
3.3.2	Activity Monitoring	43

3.3.3	Lessons Learned	44
3.4	Evaluations in this Field	45
3.4.1	Sandia Algorithm Evaluation	45
3.4.2	Point-Source Epidemic Curves	47
3.5	Sensor Placement	48
3.5.1	Genetic Algorithm	49
3.5.2	Greedy Heuristic Method	50
3.5.3	Scalable High-Performance Heuristics Method	51
3.6	Sensor Networks	53
3.6.1	What They Are	53
3.6.2	Contribution to This Problem	56
4	Methodology	59
4.1	Specify General Test Bed Architecture	59
4.1.1	Experimental Considerations	60
4.1.2	Simplifying Assumptions Made	60
4.1.3	Data Source	64
4.1.4	Initial Framework	68
4.2	Solicitation of Event Algorithm Developers	75
4.3	Finalize Evaluation Details	75
4.3.1	Feedback from Developers	75
4.3.2	Challenges	76
4.3.3	June Workshop	79

<i>CONTENTS</i>	3
4.3.4 Final Experimental Plan	86
4.4 Algorithms to be Evaluated	87
4.5 Implement Test Bed	89
4.6 Develop Simulated Data Streams	92
4.6.1 The Network	92
4.6.2 Sensor Locations	93
4.6.3 Contaminants	95
4.6.4 Attacks	98
4.7 Scoring Procedure	99
4.7.1 Definition of an Event	99
4.7.2 Detection Objectives Measured	101
4.7.3 ROC curves	104
4.7.4 DET curves	108
4.7.5 Efficiency	108
5 Results	111
6 Discussion	113
6.1 Experimental Framework	114
6.1.1 Suggested Steps	114
6.1.2 General Suggestions	124
7 Future Work	131
7.1 Continuing Evaluation Projects	132

7.1.1	WaterSentinel	132
7.1.2	TTEP	134
7.2	Testbed Improvements	135
7.2.1	ASSET	136
7.3	Data Improvements	137
7.3.1	Analysis Using Real Data	137
7.3.2	More Realistic Simulations	138
7.4	Improved Data Analysis	142
7.5	Additional Performance Measures	143
7.6	Improved Figures	148
7.6.1	Volume Under a Time-ROC Surface	148
7.6.2	AMOC curves	149
7.6.3	Histograms	150
7.6.4	Additional Plots	150
7.7	Algorithmic Improvements	151
7.8	Other Implementation Details	156
8	Summary	159
A	List of Abbreviations	173
B	File Types Used in the Evaluation	177
C	Initial Letter to Developers	185

<i>CONTENTS</i>	5
D Final Instructions to Developers	193
E Sensor Location Attacks	199

List of Figures

2.1	WaterSentinel CWS Concept Overview	14
2.2	Potential for Reduction of Fatalities Using a CWS	15
2.3	Figure showing the affect of a contaminant injection on tap water quality	17
2.4	Example of Natural Water Quality Variability	18
3.1	Key epidemic dates used in the Bio-alert study	33
3.2	Sosin's framework for the evaluation of public health surveillance systems	41
3.3	CDC's proposed process for detecting outbreaks	42
3.4	CDC's data flow plan for outbreak surveillance	43
4.1	Proposed evaluation framework	71
4.2	Map of the network used	93
4.3	Water Age Signatures of the sensors used	94
4.4	OP - sodium arsenite interaction	97
4.5	OP - chlorine interaction	98

4.6	Contamination example	100
4.7	Sample ROC curve	106
4.8	Comparison of a DET and ROC curve	109
B.1	Sample *.sim file	177
B.2	Sample *.inp file	178
B.3	*.tso file format	181
B.4	*.tao file format	182
B.5	*.out file format	183
C.1	Initial Letter sent to algorithm developers	186
D.1	Final letter sent to algorithm developers	194
E.1	Locations of simulated attacks	200
E.2	Contaminant levels at the sensors from attack 1	201
E.3	Contaminant levels at the sensors from attack 2	202

Chapter 1

Introduction

Since the terrorist attacks on September 11, 2001, the United States has been working hard to prevent future assaults and to prepare to respond quickly and efficiently if another strike should occur. One area of much recent concern has been the safety of water distribution systems. Water systems are vulnerable to attack since they are spread out over extremely large areas, have many potential access points throughout the system, and are not continuously monitored for the presence of contaminants. Contamination of a drinking water system has the potential to result in significant public health and economic impacts.

Since it is implausible to eliminate the potential for attack on large pipe networks, the consensus is that a monitoring system is needed within the water distribution systems to quickly detect in real time if contamination has occurred so that immediate action can be taken to minimize the damage done [(USEPA), 2005a]. The goal is to “place a limited number of sensors in a water distribution network such that the impact to public health due to the accidental or intentional injec-

tion of contaminant is minimized [Watson et al., 2005]. This network of sensors, including the hardware and software necessary to implement and monitor them, is here referred to as a Contamination Warning System (CWS). The official definition of a CWS is "the active deployment of monitoring technologies and/or strategies and enhanced surveillance activities to collect, integrate, analyze and communicate information to provide a timely warning of potential water contamination incidents and initiate response actions to minimize public health impacts" [(USEPA), 2005a]. Obviously, a CWS must have many interrelated components in order to accomplish these goals. A more detailed description of the proposed CWS format is discussed in section 2.1.1.

Recently, attention has been given to the analysis of information provided by network sensors. This is a vital step in developing a successful CWS: even perfect sensors placed in ideal locations are useless unless their data can be effectively processed and utilized. Previously, researchers developing a CWS operated under the simplifying assumptions that any specific contaminant could be detected, and that if the contaminant concentration at any sensor was above a specified threshold, a contamination event has occurred and would be detected. Otherwise, the network was safe. Detection was seen as a straightforward local decision. In reality, however, it can be difficult to distinguish between unsafe and safe conditions. As it is impractical to install sensors that directly monitor for each possible contaminant (as sensors are expensive, and the list of contaminants will likely continue to grow), contamination will have to be detected indirectly by monitoring standard water quality parameters, such as chlorine and pH. There are no perfect

sensors, so instead data must be mined from a set of noisy, naturally varying, imperfect time series.

This analysis of sensor data is arguably the most important step in developing a successful Early Warning Detection System: even perfect sensors placed in ideal locations are useless unless their data can be effectively processed and analyzed. Many algorithms from a variety of fields exist that could be applied to this problem, though there has been no evaluation or comparison of these algorithms to determine their effectiveness or to determine how close current algorithms come to the accuracy and reliability needed to be implemented in a CWS.

This work seeks to move the evaluation of event detection algorithms forward by establishing an experimental framework for rigorously testing event detection algorithms in a way that fairly and realistically reflects what the systems' performance would be if deployed in a water distribution system. Hopefully, the knowledge gained from this research will be useful to those who seek to evaluate event detection algorithms and will support efforts to determine the effectiveness of current algorithms and give feedback to groups working on this problem.

Chapter 2

Objectives

2.1 Motivation

2.1.1 Contamination Warning Systems

A CWS "is not merely a collection of monitors and equipment placed throughout a water system to alert of intrusion or contamination. Fundamentally, it is an exercise in information acquisition and management" [(USEPA), 2005e]. Figure 2.1 shows the working CWS diagram used in the USEPA's WaterSentinel program. The very left column of the diagram shows the possible data sources that could be part of a CWS. These include data about the actual water distribution system such as sensor data, results of periodic water sampling, operational and hydraulic data, water utility customer complaints, and plant security information, as well as data obtained from the public health sector such as over-the-counter drug sales, doctor visit results, 911 calls, and calls to poison control. Online monitoring using

water quality sensors, which is the focus of this thesis, is only piece of the larger system.

Figure 2.1: Figure taken from [(USEPA), 2005e] showing the CWS concept proposed as part of the WaterSentinel program

Once the data is collected from these varied sources, it must be analyzed to determine whether conditions are normal. The second portion of Figure 2.1 shows each of the data streams being inputted into anomaly detection algorithms to be analyzed. If the results of these analyses indicate that a credible threat has been identified, confirmation occurs and immediate action is taken to minimize the damage caused by the attack. This includes operational responses such as closing valves to shut off a portion of the system and contain the contaminant, public health responses like performing vaccinations, and risk communications such as news reports to stop drinking the water. Finally, the last column of Figure 2.1

shows that the process culminates in remediation and recovery efforts that clean the system and prepare it for public use once again.

Clearly, the implementation of an effective CWS goes beyond the data acquisition and analysis and requires the cooperation and coordination of several groups including water utility personnel, public health officials, law enforcement personnel, local officials, and the media in responding to system alarms. In the WaterSentinel System Architecture document produced by the USEPA, it is noted that while "many utilities are currently implementing some monitoring and surveillance activities, these activities are either lacking critical components or have not been in a manner sufficient to meet the primary objectives of a CWS - timely detection of a contamination incident" [(USEPA), 2005e]. However, the CWS does show promise if implemented correctly, as shown in Figure 2.2.

Figure 2.2: Figure taken from [(USEPA), 2005e] showing the potential for the reduction of fatalities in 10 simulated contamination scenarios if a CWS is implemented. The contaminant classes shown on the x-axis are described in the document.

2.1.2 Data Analysis

The CWS concept is fairly new, and researchers have only recently begun addressing the problem of sensor data analysis. In a CWS, large masses of imperfect sensor data from throughout the network must be quickly analyzed, taking into account a system's constant variability and continuously changing water quality. An effective program clearly must prevent missed detections, as damage will continue if an attack is not noticed. But false alarms are also detrimental, as extensive and intrusive containment efforts are needlessly taken and large groups of people are unnecessarily worried. This would cause resentment and skepticism towards the system and decrease the likelihood of prompt response for the next alarm.

Perfect, contaminant-specific sensors do not exist. All sensors have some measurement error. Also, there is no sensor that can directly detect all possible contaminants. Instead, the dominant CWS design concept proposes to infer the presence of a foreign substance from analysis of standard water quality measurements such as free chlorine, turbidity, and pH levels [(USEPA), 2005c]. Preliminary testing using water quality monitors to detect contamination events has been performed by Hach and the United States Environmental Protection Agency (USEPA)'s Technology Testing and Evaluation Program (TTEP). Results look promising, with all contaminants agents tested using the Hach system "caus[ing] a distinct change in one or more of the measured parameters pH, turbidity, free chlorine, conductivity and total organic carbon (TOC)" [Kroll and King, 2005]. Also, a USEPA literature review found that free chlorine, TOC, conductivity, and

pH were effective in detecting the 33 WaterSentinel 7.1.1 Baseline contaminants (those which they feel pose the biggest threat) [(USEPA), 2005d]. Figure 2.3 shows the chlorine level was affected when sodium arsenite was injected into a beaker of chlorinated Cincinnati tap water. The introduction of a contaminant clearly affects the tap water quality. Control studies were performed under identical conditions (but with no arsenite injection), and these tests showed no change in free chlorine.

Figure 2.3: Results of laboratory experiments performed at the USEPA's T&E facility. Sodium Arsenite was injected into a beaker of chlorinated Cincinnati tap water and free chlorine concentration was measured over time. Note that DPD is the standard wet chemistry method for free chlorine analysis and ATI is an electrochemical sensor.

These parameters naturally vary in a water distribution system due, in part, to fluctuating demands, pump operations, and source conditions. "Trends in water quality may be influenced by short and long-term climatic influences (e.g., diurnal cycles as well as seasonal cycles) or may be influenced by the physical operations of the distribution system (cycling of pumping wells, draining and filling of tanks).

Other trends can be caused by drift in electronic instruments measuring water quality parameters or chemical/biological processes occurring within the distribution system” [Srinivasan et al., 2005]. Figure 2.4 gives an example of how variable the water quality parameters can naturally be, as the data shown was collected during the natural operation of a pilot utility.

Figure 2.4: Figure taken from [McKenna et al., 2005] showing actual chlorine, pH, temperature, and conductivity sensor readings obtained from a monitoring station in a pilot water utility.

Interpreting these imperfect, constantly fluctuating signals is a difficult task. There are no clear-cut rules for determining whether a variation is normal or if a contaminant has been introduced into the system. Even among those in the water distribution system research and practice community, there is not a solid

understanding of the fluctuations in water quality parameters. And without a firm grasp on what constitutes “normal conditions, it will be difficult to accurately distinguish between ordinary and abnormal readings. Also, there is no historical data showing examples of attacks.

Thus, the current CWS concept requires analysis of large amounts of imperfect water quality sensor data to distinguish between normal water quality variability and that due to a foreign substance [(USEPA), 2005c, (USEPA), 2005a]. An effective algorithm must minimize missed detections, as damage will continue if a contamination event goes undetected. But false alarms are also detrimental, as response and containment efforts are expensive and the population is unnecessarily alarmed.

The need for an effective detection algorithm is clear. Private industry is selling event detection kits for water distribution systems (see [(USEPA), 2005a] for a list of vendors and developers). They often boast of high detection rates and low false positives, but the detection method and evaluation approach details are not made publicly available, and the algorithms are never compared to other systems. It is important to thoroughly evaluate detection methods so that water utilities can base purchasing decisions on objective data and analysis. Also, the USEPA is actively evaluating CWS options, through their TTEP and WaterSentinel programs [(USEPA), 2005b]. WaterSentinel plans to deploy sensors and begin collecting water quality data in a pilot utility by the end of 2006. Clearly, now is the time to develop and evaluate algorithms for event detection. Up to now, “evaluation of ... detection effectiveness in realistic settings has been min-

imal” [Buckeridge et al., 2004]. It is important to thoroughly evaluate detection methods before too many utilities install these products in their systems simply because of their availability. Clearly, now is the time to develop and evaluate algorithms for event detection.

2.2 Project Objectives

The main goal of this work is to develop and implement an experimental computer test bed for evaluating event detection algorithms for use with Contamination Warning Systems placed in water distribution systems. Actual real-world testing is impossible, as operational water systems cannot be injected with contaminants and there are currently no networks equipped with a network of optimally placed sensors like those that will be used in a CWS. Also, laboratory testing is insufficient in modeling the changing conditions and complexity of a large distribution system. Therefore, computer simulation will be used, and every effort will be taken to maximize the validity and realism under which these algorithms will be tested.

Ideally, this thesis will produce a methodology for objectively and quantitatively evaluating algorithms that that could be used to solve this event detection problem in a CWS, as well as some preliminary data on algorithm performance. The results obtained will likely help researchers improve their algorithms and move closer to an algorithm with the precision and credibility needed for a Contamination Warning System. The specific goals of this project are as follows:

- Do a thorough literature review to understand the current state of the water

quality sensor data analysis problem

- Develop an experimental framework for comparing event detection algorithms
- Develop a statistically significant, complete, non-biased analysis tool to compare algorithms for analyzing the data produced by the sensors of a CWS
- Develop the necessary software and gather the needed resources to implement this assessment tool
- Seek to establish what algorithms currently exist and make contact with as many groups as possible who are working on data analysis and event detection algorithms
- *Obtain executable code and algorithm descriptions from at least 2-3 groups*
- *Generate realistic simulation data representing both contamination events and non-events for use in testing detection algorithms. These datasets can later be made available to the public and could be used as benchmark data by researchers; alternatively, the means developed to generate them could be used by other researchers.*
- *Perform an analysis of the obtained algorithms*
- Based on experience gained, develop and refine a framework for testing detection algorithms in real-time that can be used by future researchers, and give suggestions for future work on event detection evaluations.

The italicized objectives are not yet completed. The development of benchmark data is still in progress, but the planned evaluation will be carried out once the data is available. Results will be published in another work.

2.3 Research Significance

Recently, increased attention has been given to the challenge of analyzing the constantly changing sensor data that will be produced when a CWS is installed into a water distribution system. However, up to this point, developers have used very simple methods and data to test their algorithms. For example, data has been generated by taking normal network data and multiplying each value during a given time period by some constant c to simulate an attack [McKenna et al., 2005].

I plan to set up a rigorous and thorough experiment to test different real-time event detection algorithms in order to get a better understanding of how different algorithms would perform in an actual water distribution system. As these algorithms have not yet been tested on realistic data, this will be more of a learning opportunity than a definitive test. The evaluation tool developed can be used to begin a recursive process of testing and refining. This will continue until, eventually, an algorithm will be developed that is sufficiently accurate and trustworthy to be implemented in a water distribution system. Hopefully, my thesis work will be helpful in finding out how close developers are to an acceptable data analysis algorithm, will provide tools for analyzing future work, and will push the research in this area forward.

Chapter 3

Related Work

3.1 Event Detection Algorithms

Event detection algorithms, also called anomaly detection algorithms or novelty detection algorithms, are used in a wide variety of domains. These algorithms take a time series (or multiple time series) as input and output an alarm if they determine that current data is abnormal. Some applications of event detection algorithms include trying to predict good buying or selling times in the stock market [Aggarwal, 2005], analyzing machine vibrations to detect when a part is about to fail [Zanardelli et al., 2005], monitoring credit card use for possible identity fraud [Chan et al., 1999], detecting public health outbreaks [Lippmann et al., 2000a, Haines et al., 2001a, Buckeridge et al., 2004, Burkom et al., 2004, Hutwagner et al., 2003, Burkom et al., 2005, Schmueli, 2005], and finding computer network security breaches [Siegrist and Pavlin, 2004, Bio, 2004].

There are a variety of methods used to analyze data to determine whether it

is normal or whether further investigation needs to be taken. However, the algorithms generally fall into two categories: statistical approaches and classification.

Statistical methods "involve the use of probabilities and sufficient population sample sizes to establish normal and abnormal conditions in a system, and to make predictions of future events based on the current state of the system being observed" [(USEPA), 2005a]. A simple statistical method would be to assume normal distribution and indicate that conditions are abnormal if a sensor reading falls outside of n standard deviations from the mean. However, these methods can become much more complicated, with probabilistic reasoning and more complex distributions [Markou and Singh,]. The algorithms developed at Sandia National Laboratories [McKenna et al., 2005] and the ones tested at Johns Hopkins University [Burkom et al., 2004, Hutwagner et al., 2003, Lombardo et al.,] use statistical techniques and are described more in detail in section 3.4.

Classification, which encompasses artificial intelligence (AI) techniques such as neural networks and perceptrons, involves analyzing the features of current data to identify it as normal or abnormal. Some of these algorithms are trained to "select one or more classes that have been defined in advance" [(USEPA), 2005a]. An example of this would be an algorithm that has been trained to analyze water quality data to determine whether it best characterizes normal water or one of the pre-defined contaminants. This is called supervised learning.

Clustering algorithms are similar to this in that they seek to classify current data into a category. However, with clustering, the categories are not defined in advance. They go through what is called unsupervised learning in which the

data's features are plotted in n -dimensional space, and the clusters that are formed constitute the classes. This type of training often produces non-intuitive groupings, and these can provide further insight into the nature of water quality. The ADWICE software developed at the University of Linkopings is an example of this type of classification algorithm [Burbeck and Nadjm-Tehrani, 2004]. In this algorithm, a tree of groups is built as the algorithm is trained, and this tree can be quickly searched as new data is presented to determine which, if any, of the defined categories the data falls into.

Some algorithms currently in use are described in sections 3.2, 3.4, and 4.4. This list is by no means meant to be a comprehensive list of available algorithms. Countless researchers have developed and refined their own methods for analyzing a data stream to detect irregularities.

In addition, there are some recent, useful reviews of event detection algorithms available [(USEPA), 2005a]. The USEPA has made public a document as part of its WaterSentinel program (see section 7.1.1) that discusses the development of an event detection system. A summary of event detection algorithms is included, as well as descriptions of several hardware and software tools that are being investigated for the project .

Buckeridge et al. [Buckeridge et al., 2005] describe several outbreak detection algorithms in the context of a public health surveillance system. Instead of grouping the algorithms by analysis type, this paper "provide[s] a classification of algorithms based on their functional characteristics that is meaningful to surveillance practitioners and researchers" [Buckeridge et al., 2005] and classifies

algorithms in terms of their input type: the number of data sources each algorithm can accept, the availability of covariate data, and the algorithms' utilization of spatial data.

Another excellent resource is a two-part detection algorithm review performed by Markou and Singh [Markou and Singh, , Markou and Singh, 2003], two computer scientists at the University of Exeter. The first document discusses statistical approaches to classifying a signal as normal or abnormal. Methods discussed include probabilistic approaches, Hidden Markov Models (HMMs), and hypothesis testing [Markou and Singh,]. The second part of the review discusses neural network-based approaches and includes discussions of support vector machines, self-organizing maps (SOMs), and perceptrons.

Obviously, no literature review is going to be exhaustive, as event detection algorithms continue to be developed for a variety of applications. However, these resources will at least provide a foundation for understanding the general techniques that are used, adapted, and enhanced by detection algorithm developers.

3.2 Previous Detection Algorithm Evaluations

A literature review of event detection algorithm evaluations yields numerous articles describing procedures for evaluating and comparing detection algorithms. The approaches vary in complexity and focus, with varying degrees of thoroughness and substantiation. The studies fall into two general categories: independent evaluations and self-evaluations.

The most common type of algorithm evaluation is when the algorithm developer tests the performance of their own product. These tend to be brief and incomplete studies with minimal description of the methods used. Very rarely are the results compared to those of other algorithms. These analyses tend to follow the same format: either real or simulated data are collected; the algorithm is run on the data, and results documented in terms of detections rates, false alarm rates, and sometimes detection times. Unfortunately, authors sometimes do not define the evaluation carefully; for instance, defining quantitatively what is considered to be detection. However, several good self-evaluation studies have been performed, and the work done by Sean McKenna at the Sandia National Laboratories [McKenna et al., 2005] and the evaluation of the ESSENCE tool by John Hopkins University and the National Center for Infectious Disease [Burkom et al., 2004] are described below 3.4.

This work is an example of an independent evaluation, or one that is organized and managed by an impartial evaluation group. With independent evaluations, a team of researchers who seek to compare the effectiveness of various algorithms, but who have no attachment to any particular algorithm, establish a testing methodology. This includes the development of a general testing framework, generation of testing data, and definition of performance measures. Algorithm developers then submit their algorithms for testing, following the guidelines laid out by the detection team, who then perform the evaluation and report on the results. These studies tend to be very thorough and well-documented. This section describes some of the most extensive and rigorous evaluations that have

been performed on event detection algorithms to date. Note that no independent evaluation has yet been performed in the water quality field.

3.2.1 Bio-ALIRT

The Bio-ALIRT, or Bio-Event Advanced Leading Indicator Recognition Technology, program was conducted from 2001-2004 in order to "develop the information sources, technologies and prototypes necessary for early detection of an outbreak of disease caused by the deliberate public release of a weaponized pathogen." [Bio, 2004] As part of this program, an evaluation was performed in order to test algorithms designed to detect public disease outbreaks and to determine which possible data sources (over-the-counter drug sales, school and workplace absenteeism records, and doctor visits, for example) are most informative in detecting outbreaks.

Four groups contributed algorithms to be evaluated:

- Real-Time Population Health Detector

The Real-Time Population Health Detector, developed by researchers at General Dynamics Advanced Information Systems, Inc. and Stanford University, attempts to detect disease by using a "Seasonal Outbreak Detector" and a "Sporadic Outbreak Detector." The Seasonal Outbreak Detector calculates p-values by combining a seven-day trailing average of the data with a 30-day trailing average. The Sporadic Outbreak Detector then uses p-values obtained from the Kalman filter to determine if an attack has occurred: if

the residual p-value is greater than 95%, there is considered to be an attack.

[Bio, 2004]

- IBM Algorithm

Researchers at the IBM Watson Research Center also use statistical methods for anomaly detection. Training data is aggregated and the various data streams are weighted (based on their ability provide discrimination between an outbreak and natural noise) to provide a linear combination of the data.

The algorithm then tries to predict future values by considering this weighted combination of the data from the previous days and the current day's values from previous weeks. If the current value obtained by the combining deviates from the predicted value by more than a given value, an alarm is raised.

[Bio, 2004]

- JHUAPL/WRAIR Algorithm

The third algorithm set analyzed came from the Johns Hopkins University Applied Physics Laboratory and the Walter Reed Army Institute of Research. In order to detect an epidemic, a scaled value between 0 and 1 is first obtained from two different detection algorithms. One version of the algorithm uses a multiple statistical process control (MSPC) algorithm to calculate regression residuals to deal with natural variation. It uses the total number of providers supplying records for a given day as a predictor to account for weekends, holidays, or other conditions that may produce unusual data. The second algorithm type attempts to detect anomalies statistically

by computing a moving average for each data stream and continuously calculating the baseline mean and standard deviation. Finally, a Bayes Belief Net method is used to combine the algorithm outputs to produce the final verdict on the current state. [Bio, 2004]

- Pitt Algorithms

Collaborators at the University of Pittsburgh and Carnegie Mellon University developed four different algorithms for epidemic detection. The Biosurveillance Using a Change-point Detector Algorithm (BCD) applies change-point statistics to five different parameters including the period currently being considered, the number of monitoring statistics, and the monitoring start point. The Sickness-Availability Algorithm estimates an expect count of physician visits for each day based on previous records from that day of the week and then calculates a p-value (assuming a Poisson distribution) representing the distance between the actual number of visits. The Wavelet-Based Anomaly Detection Algorithm computes a wavelet describing the data trend from the previous 32 days. It compares yesterday's residual value to the current one to compute a Z-score, which is compared to a Z-threshold to determine if an alarm should be sounded. [Zhang et al., 2003] Finally, the Simple Poisson Regression combines the data streams by choosing the maximum value or by summing the data, and then does a simple Poisson Regression (using the training data), considering the previous day's count and a variable for the day of the week. [Bio, 2004]

One major component of the study was compilation of a dataset to use in algorithm testing. Two different types of data were used:

- Simulation

In 2002, the first Bio-ALIRT algorithm evaluation was performed. For this study, the BioWar simulation tool was used to generate normal background data as well as an outbreak signal. 100 scenarios were generated that modeled the effects of an anthrax outbreak.

- Historical Data

A more extensive study was carried out in 2003. This time, authentic historical data from five cities was used. Daily records of clinical office visits and prescriptions were assembled (and cleaned so that the patients remained anonymous) for 22 months, providing three data streams (military outpatient visit reports and civilian outpatient records were gathered separately). The first 14 months of this data was used by the developers to train their algorithms, and the other eight months were used for the actual testing.

The researchers clearly preferred the second datasets that included unbiased real data, stating that "purely authentic data avoided any issues concerning the realism of the simulation" and that "use of data from cities of interest allowed the developers to hone their skills in a realistic environment." [Bio, 2004] However, the specifications of the attack are not chosen when historical data is being used, leaving uncertainty as to the exact start and stop times of the outbreaks, the sizes of the outbreaks, and the determination of whether a given time series in fact shows

normal or outbreak conditions. In order to determine when outbreaks occurred, a team of epidemiologists, called the Outbreak Detection Group, analyzed the data and identified 15 periods of time that looked like possible outbreaks that would warrant additional investigation. They did this by looking at public records to see if any outbreaks were documented in the five cities during this time, by using a data visualization tool that displayed the data sources together and allowed for grouping by city, syndrome group, and age group, and by using simple statistical methods to identify anomalies.

Once the datasets were assembled, the training data was sent to the algorithm teams, which included two labeled outbreaks. Each team had six weeks to train their algorithms for three different false alarm rates: one per two weeks, one per four weeks, and one per six weeks. The start date, date of recognition by the public, peak date, and stop date were given for each outbreak. Once training was complete, groups received the unlabeled testing data, in which the outbreaks were not identified. As the testing was performed by the actual algorithm developers, the honor code was used, and each team affirmed that their analysis processes were "repeatable and thus verifiable." [Siegrist and Pavlin, 2004] The algorithm output was collected two weeks later, and software was used to analyze the results.

Figure 3.1 shows how detections and false alarms were defined. The start date, public health recognition date, peak date, and stop date were all defined by the previously mentioned ODG team. Sensitivity, or proportion of attacks correctly identified, and timeliness were the two major metrics used to compare the algorithms. The measure for timeliness was days to detection, and the median

Figure 3.1: Figure taken from [Siegrist and Pavlin, 2004] showing the key epidemic dates and how detections were scored.

timeliness was used to compare the algorithms, with a non-detection given a detection time of ∞ . This way, all scenarios could be included in the timeliness measurement, not just the ones that were detected [Bio, 2004]. False alarms were not measured, as the algorithms were specifically trained to produce certain false alarm rates: one per two weeks, one per four weeks, and one per six weeks.

"Snapshots" of an ROC curve for the algorithms were included, as there were not enough data points to create a full curve. Activity Monitor Operating Characteristic (AMOC) curves (discussed in more detail in chapter 7.6.2) were also used which plot detection time (in days) against specificity. Other figures used to illustrate performance included a histogram showing how many events each algorithm detected in n days for each false alarm rate and charts breaking down performance for each outbreak type (gastrointestinal and respiratory) [Bio, 2004].

The results of the testing were very favorable. "The best algorithms were able to detect all of the outbreaks...at a false alarm rate of one every 2 weeks"

[Siegrist and Pavlin, 2004]. Also, the algorithms averaged at least an 18 day advantage in detection time compared to the time public officials were projected to notice the outbreak.

Several conclusions were drawn from the algorithm analysis. Off-the-shelf software is not appropriate for this problem, as too many false alarms occur. Older methods that smooth the data excessively (and thus neglect the weekly patterns that exist) do not perform well. However, general linear modeling, wavelets, and statistical process control all performed well in the testing [Bio, 2004]. Different algorithms had different strengths: different algorithms performed best when the attack fingerprint (a quick change in signal versus gradual changes), type, and false alarm rate varied. Therefore, the authors note that "this evaluation provides a 'snapshot' of the performance of certain algorithms and data-processing methods... Whether certain algorithms were better overall than others was not determined" [Siegrist and Pavlin, 2004].

3.2.2 DARPA Offline Intrusion Detection Evaluations

The Defense Advanced Research Projects Agency (DARPA) has sponsored two major intrusion detection system evaluations designed to "drive iterative performance improvements in participating systems by revealing strengths and weaknesses and helping researchers focus on eliminating them" [Lippmann et al., 2000b]. Completed in 1998 and 1999, these studies tested various algorithms' ability to detect a computer network intrusion. Many groups participated in these studies,

and extensive documentation exists. Thorough analysis of the results and findings has been performed.

Both studies used simulated data based on the network traffic at a small Air Force base. Web, telnet, and mail traffic was generated between the computers to create the normal activity. Attacks were then added into this background traffic. Thirty-two attack types were simulated in the 1998 evaluation and included denial of service attacks, remote to local attacks, user to root attacks, and surveillance / probing attacks on four different machines in the network. [Lippmann et al., 2000b]. The 1999 evaluation added 62 new attack types, including attacks on a Windows computer and new and more covert attacks to try to challenge the algorithms [Lippmann et al., 2000a].

The 1998 evaluation challenged six research groups with over 300 attacks. Eight groups participated in the 1999 evaluation, and they were confronted with over 200 attacks. Each research group first received three weeks of training data which contained both normal, background activity and some identified attacks. About two months later, they were sent two weeks of unlabeled test data on which to test their algorithm. The testing data contained several new and different attacks not present in the training set in order to test the algorithms' ability to adapt and detect new assaults. The teams were required to send back a list of suspected attacks about two weeks later, including the date and time of the alleged attack, the IP address of the suspected target, and an attack certainty score.

An attack period was defined as all time segments during which infected packets are passing through the system or during which an attack process is running

on a victim computer. Any alert returned by a research group that fell within an attack period or within 60 seconds of the end of it (extra time given because of timing inconsistencies) was considered a true positive. Anything else was a false positive. One interesting scoring decision with this evaluation was that groups submitted a list of attacks that their algorithm should identify before the testing, and they were not penalized for attacks that they were not designed to detect.

Detection was the main judging criteria in this study. ROC curves^{4.7.3} were produced for each system and for each attack category. DFA plots were created, which are similar to ROC curves except they show the probability of detection versus the false alarms per day. Also, a curve is not created but individual points are plotted [Haines et al., 2001a]. Each attack and attack category were also considered separately to see how often they were detected.

No conclusive "winner" emerged from either of the DARPA evaluations: each attack type had its own "best" algorithm for detecting it, and many different algorithms showed up in this list. As in any evaluation, the DARPA evaluators remind the reader that results "should be interpreted within the context of the test bed, background traffic, attacks, and scoring procedures used. The evaluation used a reasonable, but not exhaustive, set of attacks... It also used a simple network topology, a non-restrictive security policy, a limited number of victim machines and intrusion detection systems, stationary and low-volume background traffic, lenient scoring, and extensive instrumentation" [Lippmann et al., 2000a]

But the most unique and, arguably, the most valuable outcome of this study was the extensive evaluation of the results. To help discern why missed attack

were not detected and why alarms were raised during normal conditions, the algorithm developers were sent their results and asked to analyze them in order to answer the following questions [Haines et al., 2001a]. The first five questions were answered for each attack that was missed, and the last two were considered for any false alarms raised.

1. Was this attack within the specification of this intrusion detection system?

Researchers were asked if they felt their algorithm should have detected the given attack.

2. Was evidence of the attack "visible" in the data used by this intrusion detection system?

Some of the evaluated algorithms used only a subset of the available data in their calculations. This question asks whether the attack was present in those data streams that were considered.

3. Did the "features" extracted from the input data preserve evidence of the attack?

Again, this question seeks to determine if the attack should have been detected based on the method in which the algorithm analyzes the data.

4. If the attack was within the specification of the system and input features contained evidence of the attack, why wasn't it detected?

Possible answers here included a bug in the program, a belief that the evaluators incorrectly marked an attack, the fact that it is a new kind of attack,

or an unknown cause.

5. How hard would it be to extend the intrusion detection system to detect this attack?

Developers were to indicate if they could modify their algorithm to detect the attack within a day, week, month, year, or if it would be impossible.

6. Why did this false alarm occur?

Answers were similar to those from question four.

7. How hard would it be to extend the intrusion detection system to eliminate this false alarm?

See the answers from question five.

This in-depth analysis helped the developers and evaluators fully understand the data and the workings of the algorithms. Weaknesses and bugs of the algorithms were revealed, and attacks that could not currently be detected were clearly defined. Overall, the algorithms performed poorly on new attacks. Clearly, the ability to adapt needs to be developed, as hackers are constantly developing new attack strategies.

Since the 1999 DARPA evaluation, several extensions have been made to improve testing capabilities. First, two additional datasets have been created. These are shorter simulations on a more complex network than the one used in the original evaluations. This data has been made available to researchers for testing purposes.

A new, flexible software tool for algorithm testing has been developed. The Lincoln Adaptable Real-Time Information Assurance Testbed (LARIAT) "provides easily configured versions of the background traffic generation and attack scripting software used in the 1999 [DARPA] evaluation and a graphical user interface (GUI) for configuration and control" [Haines et al., 2001b]. The testing environment contains a network director that controls the testbed, traffic generators that simulate network hosts, and attackers that launch attacks.

Finally, the evaluators note that "a thorough understanding of the input data streams, features, detection algorithms, and output helps in understanding what evidence must exist for an intrusion detection system to detect an attack" [Haines et al., 2001b]. The insights gained in these areas from the analysis of the DARPA evaluations is being used to develop Feature-Analysis models. These models can be used to characterize detection systems by the input features they use and their analysis methods and to classify attacks by the evidence they leave. This can be useful in predicting an algorithm's detection capabilities, planning future evaluations, and producing more realistic alert streams.

3.3 Proposed Evaluation Frameworks in Other Fields

As seen in section 3.2, extensive evaluations have been performed in other detection fields. As a result of these assessment experiences, guidelines for effectively

comparing event detection algorithm performance have been developed for these fields.

3.3.1 CDC Recommendations

The Center for Disease Control and Prevention (CDC) expanded on the framework developed by Daniel Sosin [Sosin, 2003] for assessing surveillance systems designed to detect disease outbreaks (see figure 3.2). Such a framework is needed "to establish the relative value of different approaches and to provide information needed to improve their efficacy for detection of outbreaks at the earliest stages" [James W. Buehler et al., 2004].

The first step in this proposed framework is to fully describe the system, including its purpose, stakeholders, and operational details, such as data processing, statistical analysis tools, and decision-making procedures. Flow charts, such as the ones proposed in figures 3.3 and 3.4 would be effective in representing this information.

Once all stakeholders agree on the functionality and intended uses of the system, its ability to detect outbreaks must be addressed. Several measurements of effectiveness are proposed in this framework. Timeliness is defined as the "lapse of time from exposure to the disease agent to the initiation of a public health intervention" [James W. Buehler et al., 2004]. This includes the latency period between exposure and symptom appearance, the time necessary to collect and process the data, running time of the event detection algorithms, and the

Figure 3.2: Figure taken from [Sosin, 2003] outlining a framework for evaluation of health surveillance systems.

TABLE. Framework outline	
A. System description	
1. Purpose:	What is the system designed to accomplish?
2. Stakeholders:	Who is the system serving?
3. Operation:	How does the system work?
a. Use by stakeholders	
b. Case definitions	
c. Process model	
d. Data model	
e. Interoperability	
f. Detection algorithm	
g. Privacy/confidentiality	
h. Communication	
B. Experience	
1. Usefulness:	In what ways has the system demonstrated value relevant to public health?
2. Acceptability:	Have stakeholders been willing to contribute to and use the system?
3. Generalizability:	How readily can the system be duplicated in another location?
4. Stability:	How consistent has the system been in providing access to reproducible results?
5. Cost:	What are the resource requirements to deploy and maintain the system?
C. Outbreak detection	
1. Flexibility:	How adaptable is the system to changing needs and risk thresholds?
2. Sensitivity and predictive value positive:	What proportion of true cases and outbreaks are detected by the system? What proportion of alarms triggered by the system are desired alarms (true positives)?
3. Timeliness:	How early in the disease process or outbreak is the event detected?
D. Data quality	
1. Representativeness:	How well does the system reflect the population of interest?
2. Completeness:	What percentage of data is present for each record?
3. Reliability:	Are data captured consistently across the system and over time?
E. Conclusions and recommendations for use and improvement of the syndromic surveillance system	

time needed to analyze the alert signal, contact public health officials, and begin intervention.

The validity of a system, or its ability to accurately detect an outbreak, is the other piece of detection analysis. This requires extensive testing of the systems to measure outbreaks detected, false alarms, and outbreaks missed. Acceptable values of false positives must be considered, and "sensitivity and [predicted false positive rates] for these surveillance systems will ultimately be calibrated in each system to balance the secondary benefits... with the locally acceptable level of false alarms" [James W. Buehler et al., 2004]. A direct comparison is needed and

Figure 3.3: Figure taken from [James W. Buehler et al., 2004] illustrating the proposed process for detecting outbreaks.

simulation is the recommended method for generating a common test set (as an abundance of thorough, historical data is not available). If real data is used, the representativeness, or how well the data describes the normal distribution of cases in the given area, as well as the completeness of the data being used must be reported.

The CDC believes it is important to report on the experiences had during the testing process. They suggest describing the following system characteristics: usefulness, or how much value would be added if the system were in place; flexibility, or ability to adapt as system usages change or more data about trends is obtained; acceptability, or feasibility in terms of the stakeholders' willingness to cooperate and provide necessary data; portability, or ease in applying to another setting; stability, or resilience to change without extensive recalibration or service; and system costs, including equipment needed, personnel to run it, running costs,

Figure 3.4: Figure taken from [James W. Buehler et al., 2004] showing the proposed data flow in an outbreak surveillance system.

and the cost of investigating alerts.

Finally, conclusions and suggestions should be made at the end of any study. Strengths and weaknesses of the system should be noted. Recommendations for future improvements or continued consideration of the system should be made. Hopefully, better health surveillance systems will be developed if these guidelines are followed.

3.3.2 Activity Monitoring

Fawcett and Provost [Fawcett and Provost, 1999] propose a new class of problems: activity monitoring problems. They define this class as problems where large quantities of data are processed to look "for interesting events requiring action" [Fawcett and Provost, 1999]. This includes problems in countless fields, and they cite the examples of detecting cellular phone fraud, predicting hot stocks based on news stories, and computer network intrusion detection. The input data streams

for these problems can be anything from numbers to text to vectors of data. But no matter what the input, the goal is to "issue alarms accurately and in a timely fashion" [Fawcett and Provost, 1999] whenever unusual data is seen.

A framework is proposed for evaluating any problem from this class. For each detection approach, two functions are defined: a score function, and a false alarm function. These are user defined functions that take the event start time, detection time, and a portion of the data stream as input and output either the benefit of the given alarm on the specified sequence or the cost of the false alarm. The results are used to construct AMOC curves (see section 7.6.2, an evaluation visualization tool they describe specifically for this problem set.

3.3.3 Lessons Learned

The work and suggestions from evaluations of event detection algorithms in other fields can be extremely useful. As Fawcett and Provost [Fawcett and Provost, 1999] observe, the problem of detecting unusual conditions in a water distribution system is essentially the same as data processing tasks in other fields^{3.3.2}. The performance measures and goals are the same whether the algorithm input is public health data, packets sent across a computer network, or water quality readings.

Detection of water quality abnormalities is especially similar to the problem of detecting public health epidemics, as data is temporally and spatially correlated and background conditions naturally vary throughout the days, weeks, and seasons. The development of this evaluation framework was begun before a thorough

literature review of work in the public health surveillance field was performed. We later found our plans to be very closely aligned with previous evaluations 3.2 and suggested framework from this field. This certainly bolstered confidence in this proposal, and additional ideas were also generated by studying the work done in the public health field.

3.4 Evaluations in this Field

As noted previously 3.2, analysis of algorithms designed to detect events in water distribution systems has not nearly been as extensive as the evaluation studies performed in the public health and computer network domains. However, some notable work has been done, and two quality studies are described here.

3.4.1 Sandia Algorithm Evaluation

Sandia National Laboratories has developed two event detection algorithms specifically for water distribution system sensor analysis. These algorithms have been extensively tested using a combination of real and simulated data.

Data from two water utilities was used to test the data. The first utility measured total chlorine, pH, temperature, and electrical conductivity at 22 locations within the network each hour. Preprocessing was done with both datasets to eliminate missing values and remove extra measurements that occurred. The second utility collected measurements every 15 minutes at three stations. Dissolved oxygen, pH, specific conductance, temperature, and turbidity data were available

for this network.

The first method proposed to analyze water quality data was a linear prediction algorithm. This approach "utilizes a linear predictor to estimate a sensors' current measurement based on past observations" [McKenna et al., 2005]. This predicted value is then compared to the actual value obtained from the sensor. If the difference between the predicted and observed values is above some specified threshold, an alarm is raised.

The other method is to use time series increments, or lag-one differences. This is similar to the previous method in that the current observation is compared to a predicted value, and if the difference between these two is greater than some defined value, an attack is inferred. This method is much simpler than the previous, however. Instead of using the linear predictor, which looked at a whole sequence of past measurements and performed a series of calculations, the predicted value here is simply the measurement from the previous timestep. While the linear predictor method sought a deviation from the normal pattern, the time series increments algorithm looks for a change from one timestep to the next.

A third change detection approach, based on multivariate classification and discrimination, was also mentioned. This algorithm has not been fully developed, but the idea is to classify a finite set of water states. New data is then compared to these determined classifications, and if the measurements do not fit into an existing water type, they are thought to be caused by an attack.

To evaluate the linear filtering and lag-one differences algorithms' performance, 30 random periods in the data were chosen and the data was altered to simulate

an attack. This was done by multiplying the original data value within this interval by four different "spike strengths": 1.0, 2.0, 3.0, and 4.0. ROC curves (see chapter 4.7.3) were constructed for each spike strength level by varying the threshold which determined the alarm. The results were very promising. By combining the two algorithms together, "it may be possible to tune the algorithm to operate at or above 90 percent detection with a false alarm rate of less than 1 or 2 percent" [McKenna et al., 2005]. It was noted, however, that this method of manually adding attacks into the data does not necessarily accurately represent the behavior or an actual attack.

3.4.2 Point-Source Epidemic Curves

Researchers at the John Hopkins University Applied Physics Laboratory and the Bioterrorism Preparedness and Response Program collaborated to devise a way to test two algorithms designed to detect public disease outbreaks [Burkom et al., 2004]. The first algorithm comes from the CDC Early Aberration Reporting System (EARS) [Hutwagner et al., 2003] and uses the CUSUM formula to detect abnormally high levels of reported illness by sending an alarm if the current count is more than n standard deviations greater than the mean from the previous m days, where n and m are varied to produce different results. The second algorithm calculates an exponential weighted moving average to determine if a signal exceeds the designated threshold and is taken from the ESSENCE biosurveillance system [Hutwagner et al., 2003, Lombardo et al.,].

Simulated data was used in this study, and the background values were obtained from an online simulated database available at [http : //www.bt.cdc.gov/surveillance/ears/datasets.asp](http://www.bt.cdc.gov/surveillance/ears/datasets.asp). Artificial outbreaks were created using several epicurves, which were injected into the background data at random start times. These epicurves were time series of the cases attributable to the outbreak and the values were considered to be proportional to the epidemic curve. Specific details on how these were created are available at [Burkom et al., 2005].

ROC and AMOC curves were then generated to visualize the algorithms' performance. These were constructed in the usual way (see 7.6.2 and 4.7.3). Timeliness was measured in days and could be measured precisely because the epidemics were manually added.

3.5 Sensor Placement

Effective event detection analysis is just one component of a successful EWS. In fact, even the most perfect algorithm is worthless without informative, accurate data to analyze. In the past few years, many groups have attempted to determine how to most effectively deploy sensors in a given network, seeking to "place a limited number of perfect sensors in the pipes or junctions of a water network so as to minimize the expected amount of damage to the public before detection" [Berry et al., 2004]. They differ in both the methods used (which include exact MIP solvers and heuristic algorithms that are not necessarily optimal) and in

what parameter they seek to optimize (from the number of vulnerable nodes to the amount of contaminated water consumed to the health impact). This chapter summarizes some of the top methods that have been developed since 2001.

3.5.1 Genetic Algorithm

Ostfeld and Salomons [Ostfeld and Salomons, 2003], [Ostfeld and Salomons, 2004b], [Ostfeld and Salomons, 2004a] continue to develop and report on their method that uses a genetic algorithm to locate sensors. Their goal is to find the optimal layout of an early warning detection system (EWDS), or series of monitoring stations, so that a contamination can be perceived in enough time to take steps to reduce the system corruption. Their method has grown in complexity to consider the randomness of the quantity of contaminant discharged, random demands and unsteady hydraulics, and the decay of a contaminant as it moves through the system. To solve the sensor problem, they first create a Randomized Pollution Matrix (RPM) that includes information about the network, contaminated nodes, and sensor detections. Then their software, optiMonitor, uses a genetic algorithm to analyze this matrix and find the optimal sensor placement. Their objective is to maximize the percentage of attacks that are detected.

One aspect of the problem that only they address is redundancy. They evaluate their results to see how often multiple sensors simultaneously recognize an attack and try to reduce this to eliminate unnecessary sensors. However, genetic algorithms are not provably optimal, and this algorithm has only been tested on

reasonably small networks.

3.5.2 Greedy Heuristic Method

Uber, Janke, Murray, and Meyer [Uber et al., 2004] have recently developed a heuristic method for sensor placement. It is scalable to large networks and has been applied to networks having up to (fill in) nodes. Also, it "properly accounts for uncertainty in threat scenario characteristics and fate and transport within the distribution network" [Uber et al., 2005].

Sensor placement is based on the results of Monte Carlo simulations of different threat scenarios generated using the Threat Ensemble Vulnerability Analysis Data Generator program (TEVA). A threat scenario is "a joint probability density function of random variables that describe[s] a contamination event" [Uber et al., 2005]. TEVA utilizes EPANET to obtain detailed and realistic data on the fate and path of contaminants.

These results are used with a greedy heuristic method to find optimal sensor placement by maximizing the function

$$\sum_{i,k} p_i q_k y_i^k \quad (3.1)$$

$$\sum_k q_k = 1 \quad (3.2)$$

where p_i is the population of node i (estimated by the water demand at the node), q_k is the probability of scenario k occurring, and $y_i^k \in \{0, 1\}$ depending on whether

node i is protected by a sensor for scenario k . The goal is to minimize the public health impact of an attack, which is why population is the measurement factor used in their calculations. Their measures of health impact include the number of people susceptible to contamination (who drink contaminated water), the number of people infected by the attack, the number who become diseased, the number who die, and the dose of contaminant that the average person ingests. Probability coefficients are used to estimate the number of people at each contaminated node that will have the various reactions.

Their calculations are based on the assumption that once contamination has been detected, there will be a perfect system-wide warning resulting in a perfect response so that no negative effects will continue after the detection point. Also, they assume that detection is a purely local decision and do not consider coordination between sensors [Uber et al., 2005]. The "detection time" for a given scenario is simply the smallest time t for which one of the sensors nodes in the network detects a non-zero level of contaminant.

3.5.3 Scalable High-Performance Heuristics Method

Watson, Hart, Berry contend that their most recent work is ground-breaking as it is the "first demonstration of truly scalable algorithm performance for heuristic sensor placement optimization" [Watson et al., 2005]. They note that many previous methods suffer from one or more of the following flaws: they are tested only on unrealistically small networks and are not scalable, there are no performance

bounds and results are simply based on a heuristic, and extensive assumptions are made that over-simplify the problem. This method finds exact optimal solutions for networks having up to 3000 nodes. And it uses a heuristic to find solutions to larger networks which have also been proven to be optimal.

The objective of this method is to minimize the volume of contaminated water that is consumed. Like Uber et al. [Uber et al., 2004], they assume perfect sensors and perfect action when an attack is detected. Also, they take detection to be a local decision and use EPANET to obtain detailed water quality data. This method uses the Greedy Randomized Adaptive Search Procedure (GRASP) to efficiently find solutions, a method originally developed by Resende and Werneck [Resende and Werneck, 2004] to position facilities in industry and business. This is a heuristic approach and uses hill-climbing methods to find locally optimal values.

But the key to their work is that they also solve the problem using a mixed-integer program solver (MIP). MIP solvers solve the problem exactly, and therefore guarantee the optimal result. The time and memory needed to run these programs are overwhelming even for small networks. However, solving with an MIP solver allowed the developers to prove that their heuristic program did indeed find optimal solutions for the networks they tested. Their p-Median method finds solutions quickly and efficiently, working 15 to 30 times faster than MIP solvers and using only 10% of the memory.

This is the first time that a heuristic solver has been proven to provide optimal results for this type of problem. Also, this is the first algorithm that demonstrates

scalability, or the capability of producing optimal results for increasingly larger networks. They hope that their "results serve as a yardstick for future research on algorithms for sensor placement", and "emphasize the necessity for moving beyond "toy" sized test networks" [Watson et al., 2005]. However, they note that one major obstacle that still must be overcome is the cumbersome initial water quality simulations.

3.6 Sensor Networks

3.6.1 What They Are

In the past several years, the term "sensor network" has come to mean something very specific which does not describe the systems of sensors proposed for use in water distribution systems. Li, Son, and Stankovic define sensor networks in this way: "Sensor networks are large-scale wireless networks that consist of numerous sensor and actuator nodes used to monitor and interact with physical environments... They store environmental data on distributed nodes and respond to aperiodic and long-lived periodic queries" [Li et al., 2003a]. Sensor networks often contain thousands or even millions of small, inexpensive sensors that are densely deployed in the area of interest. Because of the sheer number of sensors, their positions are usually not pre-determined. Instead, large numbers of sensors are usually randomly scattered or even dropped from an airplane or delivered by a missile [Akyildiz et al., 2002a]. They then use wireless communication to organize

themselves into clusters and establish contact with the "base".

The actual sensors used in sensor networks are generally very small and very inexpensive (the goal is to make sensors that cost under \$1). Each sensor contains a sensing unit, a processing unit, a communication unit (for both sending and receiving data), and a power source. Some may be more complex and include extra features such as actuators or GPS systems. Because of the low cost and the fact that they are often deployed in dangerous or unreachable areas, sensors in a sensor network are usually dropped and forgotten. There is no maintenance and batteries are not changed. However, because of the sheer magnitude and density of sensors, the network can still function well if one or more sensors is damaged [Akyildiz et al., 2002a, Akyildiz et al., 2002a].

Understandably, organizing, managing, and obtaining useful from such a large collection of sensors has many challenges. I will describe some of the major issues being worked on today below.

- **Power management:** Since there is no up-keep of the sensors, the sensor network is done once the power sources on the sensors run out. As a result, then, power management is a huge area of research in sensor networks. Sensors use power for three things: sensing, processing, and communicating. In the first area, researchers are trying to determine if sensors really need to be active at all times or if sensors can cycle between active and sleeping states [Ren et al., 2005]. To minimize the amount of energy used for processing, work is being done to send queries only to those sensors that

are likely to be affected by the query and to only do a full analysis when a significant change occurs in the sensed data [Dutta et al., 2005]. And finally, to reduce the power used in communication, researchers are trying to make it so that much of the processing is done by the individual sensor and only relevant information is sent [Li et al., 2002] the shortest distance possible [Akyildiz et al., 2002a].

- **Wireless Communication:** Since the sensors in a sensor network are wireless, communication must be done through either radio, infrared, or optical means [Akyildiz et al., 2002a]. This causes problems including interference between sensor signals (which will certainly occur when there are so many sensors), data from different sensors arriving at different times, lost or damaged transmissions, and the sheer volume of data being transmitted. Much research is being done in effective routing, clustering, time synchronization, and communication management [Akyildiz et al., 2002b, Estrin et al., 1999].
- **Sensor Errors / Failure:** With small, inexpensive sensors that are not monitored, there will certainly be damaged or inoperable sensors in the network. This is in addition to the normal variability and errors associated with any sensor. A sensor network must be able to adapt to these problems. Research is currently being done in reorganization after a sensor failure, as well as in developing methods for "distinguishing between faulty sensor measurements and unusual environmental conditions" [Krishnamachari and Iyengar, 2004].

- **Query Versatility:** The main goal of sensor networks is to be able to send a query to the network and receive a correct, timely reply. Research is being done in managing queries (where to send them, how long they should be in effect,...) as well as in developing software that allows increasingly complex queries to be written [Abadi et al., 2005, Li et al., 2003b, Jiao,].
- **Accuracy:** A sensor network is used with the expectation that it will give accurate and useful information. Clearly, the network should detect the majority of events that occur but have a low number of false positives. Work is currently being done on analyzing results to establish confidence in what the network reports [Krishnamachari and Iyengar, 2004].
- **Security:** Security is certainly a major issue when information is being sent wirelessly in a sensor network. Appropriate encoding and safety measures will certainly need to be incorporated into any sensor network.

3.6.2 Contribution to This Problem

The Early Warning Detection Systems currently being developed assume a limited number of sensors which are tied into a central processing unit. This is very different than the sensor networks described here. Though it can be considered, it seems unlikely that this type of large wireless sensor network will be placed in a water distribution system. Likely difficulties would include the positioning and securing of sensors in pipes with constant water flow and the problem of getting the sensors out of the network once they are no longer functioning. Also, detection

is more difficult in this case and is not an easy query that can be processed and answered simply by an on-sensor processor. So data would still need to be transmitted to a central hub for processing, which would likely be difficult as a transmission would have to be made from inside an underground pipe.

In addition to these physical differences, the problems being tackled are simpler than the one discussed here. For all of the applications described, detection rules could be written in query form. When trying to detect a phenomenon, it was clear what to look for (for example, when detecting fires it is obvious to see when the heat sensor gets past a certain temperature) and usually detection could be done within a single sensor. Our detection problem is much more difficult, however, and there are not such clear rules to define.

However, a look at these sensor networks is helpful when looking at placement of sensors in the problem described above. Many of the issues being dealt with by those working on these networks including power management, routing, sensor errors, and confidence will need be addressed before a CWS can be implemented. Also, the work on processing data, communicating, and verifying results may be applicable to this more difficult problem.

Chapter 4

Methodology

In this chapter I will describe the steps taken and my experiences working on this project.

4.1 Specify General Test Bed Architecture

This thesis work begin in mid-September 2005 when objectives were defined and a general strategy was laid out. The scope of the project was very unclear at the start. Original ideas that were later discarded because of the lack of time, resources, and expertise included the development of a new event detection algorithm and the development of a real-time supervisory control and data acquisition (SCADA)-based interface. The outcome of those initial dialogues is included here.

4.1.1 Experimental Considerations

There are many articles that describe the lack of thorough and well-defined experimental work in analyzing algorithm performance [Flexer, , Johnson, 2002]. Flexer [Johnson, 2002] notes that “even top journal publications are possible without meeting even the simplest statistical standards of experimentation. Johnson states that “in many ways the implementation of an algorithm is the easy part and that “the hard part is successfully using that implementation to produce meaningful and valuable...research results [Flexer,].

Similar observations were made as evaluation literature was reviews for this work. Most evaluations were performed by the algorithm developers themselves, and often the results were reported with very few specifics. For example, the probability of detection was stated without any indication of how it was calculated. Also, details about the testbed, such as data generation and the algorithm’s method of accessing the data, were rarely given.

In light of these realities, the goal of this research is to develop a thorough, statistically sound experimental design that produces significant results. These guidelines, as well as insights gained from the implementation of the experiment, can hopefully be used by future researchers.

4.1.2 Simplifying Assumptions Made

There were countless ideas thrown out when ideas were be generated for this thesis work. To address them all would take years, and would require time and resources

far beyond the scope of the work. However, it was noted that this was just an initial attempt at evaluating event detection algorithms that use water quality data, and so from the very beginning, several simplifications were made to make the problem tractable.

Focus on One Network

Though it was suggested to generate data from several EPANET networks, the decision was made to consider only one network (described in chapter 4.6.1). This would simplify both the training process and evaluation.

Consideration of Each Sensor Separately

It was decided that the algorithms would be asked to analyze the time series from each sensor separately to find the local contaminant presence, instead of asking them to analyze all sensor signals to determine a system-wide status. The fusion of data from multiple sensors is common and successful in sensor networks where thousands of sensors are placed very close to each other and their correlation is evident. However, in a CWS, there will likely be only a few sensors spread out over many square miles. Changes seen by one sensor will likely not be sensed by another in a short amount of time. In fact, it is difficult to define the relationship between two sensors, as hydraulics and flows are constantly changing; thus successful fusion of sensor data may need to involve additional information about network hydraulics and operation. So at one time, sensor *a* may see a contaminant five hours before sensor *b*, but open a pump and now water first

passes sensor b and then travels to sensor a in an hour. Perhaps the synthesis of multiple signals will be considered in future studies, but it seemed reasonable and sufficient to focus on detecting abnormalities in a single stream in this initial study.

Chlorine Sensors:

Sensors that measure a variety of water quality parameters have been suggested for use in a CWS including pH, conductivity, chlorine, turbidity, dissolved oxygen, and TOC [Jonkergouw et al., 2004, Kroll, 2005]. These can be analyzed separately or in combination. In this study a single sensor is assumed to measure free chlorine concentration at each monitoring location. This assumption greatly simplifies the test bed design, as well as the generation of simulated event data. Free chlorine was chosen as the parameter to measure for several reasons. As in the decision to analyze each sensor's time series separately, it was concluded that only one water quality measurement would be used for this study.

Chlorine was chosen as the parameter to measure for several reasons. This decision simplified the test bed, made the MS-Epanet simulations more reasonable, and ensured that the evaluation truly measured an algorithm's ability to find anomalies in a data stream. Quality chlorine sensors have been developed and are already in use in many water distribution systems to monitor chlorine levels to optimize chlorine dosages, making them a natural choice for use in a CWS. Also, "most water contaminants will consume FAC (free available chlorine)," [Jonkergouw et al., 2004] and the drop in chlorine that this causes can be used to detect an attack. In fact, laboratory tests have shown that "chlorine [is] the

most sensitive water quality parameter, showing significant changes from baseline” when a variety of contaminants are introduced [(USEPA), 2005d]. Finally, on a more practical note, chemical reaction kinetic models (necessary to generate the simulated event data) are more readily available to simulate the interaction between chlorine and potential contaminants.

Limited Attack Types:

As described earlier, the DARPA evaluations 3.2.2 sought to challenge the algorithms with as many different and varied attacks as possible, including ones not in the training set. However, since this is just an initial attempt at algorithm evaluation, attacks are limited to a single continuous injection of two different contaminants. The specifics of the attack (location, start time, duration, and mass injection rate) were varied to produce different scenarios4.1.4

Focus on Algorithm Performance. Algorithm analysis focuses on two performance characteristics: correctness and efficiency. Correctness measures an algorithm’s ability to produce the correct output (in this case, detect attacks quickly while not producing too many false positives). Efficiency deals with measures such as speed, memory usage, reliability, ease of use, and running cost. For this evaluation, it was decided to focus primarily on correctness. Accurate detection is the main purpose of a CWS, and effective algorithms can likely be fine-tuned to make them more efficient.

Perfect Sensors and Transmission

Section 3.6 discusses many of the issues that arise when collecting data from sensors including missing data, sensor error, and transmission error. While these concerns will need to be addressed in the future, here the algorithms are given perfect data for training and testing: all data is available and accurate. This simplification is reasonable because none of the algorithms being tested have yet addressed the issue of missing or imperfect data, and again this is simply a first attempt at this type of evaluation.

4.1.3 Data Source

Algorithm evaluations use one of three data types: authentic historical data, historical data with embedded simulated attacks, or purely simulated data. Each of the data sources has its pros and cons. Depending on the performance measure that is being investigated, different data sources may be appropriate.

Historical data can be obtained from laboratory testing or from a water utility. Laboratory data, such as that being collected at the USEPA's Testing and Evaluation Facility's by injecting contaminants in a closed loop and observing the resulting water quality changes, has also been suggested as a source for real data. However, this data does not reflect the complexity and constantly changing conditions that are inherent in a water distribution system.

Actual data from water quality sensors in the field is obviously desirable, since it intrinsically shows real water quality conditions in a water distribution system.

It also accurately reflect the quality of data that will actually be fed to the algorithms deployed in a CWS, with missing and faulty data as well as sensor error included.

In order to be suitable for a thorough evaluation, a dataset should include a large collection of sensor signals under normal conditions as well as under contamination events. However, as most water distribution systems have not experienced a contamination incident, much less collected water quality data during the episode, no historical water quality data showing conditions during a contamination incident exist. In addition, even if an incident had occurred, the data is likely not thorough enough to be able to definitively indicate when the attack began and ended. The development of such a dataset is practically impossible; indeed the only plausible way to develop such a dataset would be to intentionally contaminate a real system in actual operation – multiple times. This clearly cannot be done. So, while this authentic data may be useful in training or in eliminating false positives, they provide no opportunity for detection.

Some researchers [McKenna et al., 2005] have tried to remedy this problem by taking historical water quality data a water utility and superimposing an event signature into the data stream (in the form of spikes, waves, or other forms representative of changes witnessed in laboratory experiments). Such a procedure is attractive because the background signal represents actual field conditions. However, no realistic information about the contaminant or its reaction dynamics is incorporated into the data using this approach, and the superimposed event at best mimics the ideal changes in sensor signals measured under laboratory con-

ditions. Thus, the simulated event data does not reflect the fate and transport processes operating in a real water distribution network, including transport and mixing through multiple paths with different residence times, and the potential kinetic effects of chemical reaction dynamics.

The third approach is to use simulated data. There are varying degrees of complexity depending on the data generation models, and this obviously affects the validity of the data generated. However, simulated data can be very useful if it is generated in such a way that it truly reflects a utility's water quality and accurately models the contaminant. Though the danger of biased or unrepresentative data is always present, methods are now being developed that produce realistic data streams and incorporate information about the hydraulic path and reaction characteristics of the injected contaminant. A major advantage of this method is that the data generated is representative of what would occur in the distribution system. Since the attack is implicitly defined, the exact start and stop times of the event are known, and the identity and concentration of the contaminant are known. Also, an unlimited amount of data can be generated in this way. A drawback of this approach, however, is that it is not representative of the challenges, such as erroneous signals, that can occur in the field from unreliable sensors and unproven technology.

It was clear that a large quantity of realistic chlorine data, both showing normal conditions and circumstances with contaminant present, was needed for this evaluation to be carried out and for the results to be credible. However, this type of data is not currently available. Field data that has been collected is largely

incomplete, and there is no data that contains a known contamination event.

Therefore, computer simulation using multi-species network water quality models was chosen to generate the event and non-event data for this study. This is a flexible approach that, in principle, can reflect important network fate and transport mechanisms. This simulation method is challenged, however, to accurately simulate variable water demands, source water quality conditions, and other exogenous factors responsible for true background variability. More work remains to be done in this area to generate truly representative datasets for evaluation, but it is believed that the data produced by the described generation method will be a significant advancement over previous approaches for several reasons.

The network simulation model relies on MS-Epanet, a multi-species extension of the widely-used hydraulic and water quality simulation model, EPANET. MS-Epanet is driven by contaminant reaction dynamics models that are specific to the assumed contaminants. The reaction and decay equations used are based on data published in the open literature as well as experiments conducted by the USEPA to predict interactions between contaminants and water quality indices if simulated contaminants were injected into a water distribution system.

MCSim is used to model the network's natural and inevitable variability. MCSim is a Monte-Carlo simulation environment that will generate stochastic simulations of contamination events by considering uncertainty and variability in key model parameters and processes such as water usage rates, tank operations, and contamination scenario characteristics. The following parameters will be varied to defining the Monte Carlo simulation ensemble: nodal demands, source chlorine

concentration, system pump operation, and contamination attack parameters (duration, start time, location, and mass injection rate). Different contaminants will be used to represent different interactions of contaminant with free chlorine residuals.

We believe that the data generated for this project using the software tool described below will be the most realistic and extensive dataset that has been generated and used for algorithm analysis. The data for this analysis will be generated by computer using network water quality simulation models, established fate and transport models predicting interactions between contaminants and water quality indices, and a Monte-Carlo simulation framework to represent uncertainty and variability in key model parameters such as water usage rates and contamination scenario characteristics. This is the first known time that a specific contaminant and its properties have been explicitly incorporated into synthetic data generation, which should give a better idea of how a contaminant injection will actually manifest at the sensors. Though there are always difficulties with simulation and evaluation, this data will be flexible and accurately reflects actual network water quality conditions.

4.1.4 Initial Framework

The main consideration when developing the experimental test bed was to simulate, as accurately as possible, the conditions in a water distribution system under which the detection algorithms will actually be run. The following describes the

initial experimental plan.

Though the algorithms would ideally be tested in real-time using the SCADA system already used by water utilities, this SCADA simulation approach was beyond the scope of the present effort (though it is currently being developed - see 7.2.1). Therefore, it was decided that all algorithms would be tested using the same one-week scenarios generated using the software tool described above. Each would contain at most one simulated attack. 50 weeks of training data would be provided to the algorithm developers, and then the algorithms would be tested by the evaluators at the University of Cincinnati using 50 weeks of test data, generated in the same way as the training data. The attacks would have random features including

- Duration: proposed to be a whole number between 0 and 9 representing the number of hours of the continuous injection of contaminant
- Start Time: a whole number between 0 and 168 (hours in a week) representing the number of hours since the beginning of the simulation that the injection begins
- Source: a whole number between 1 and the number of nodes in the network giving the random node where the injection takes place
- Injection Rate: a whole number specifying how quickly the contaminant is injected
- Contaminant: either 0 or 1 to indicate which of the two considered conta-

minants is injected

At each timestep, the algorithms would be required to output whether network conditions are normal or abnormal, and these outputs would be used to determine the effectiveness of the algorithms.

Figure 4.1 shows the proposed evaluation setup. Note that because of the positive feedback received, this framework was not changed and describes the final experimental plan. In the figure, items enclosed in a box represent a piece of software, items in a cylinders are file types, and the arrows indicate data flow. Also, items outlined in bold have been developed as a part of this project, with the shaded box being software developed for this evaluation. The box outlined with a dotted line represents the software to be tested which will be provided by the algorithm developers.

The top third of figure 4.1 shows the steps required for the data generation described above. The middle shows the data processing necessary to get the data in the format needed to input into the event detection algorithms. The actual execution of the algorithms, collection of output, and the reporting and analysis of results is highlighted in the bottom of the figure. A description of each component is given below. A sample of each file type is included in Appendix B.

- **Simulation Specification File (**.sim*)**: This file is an input into the MCSim Simulation engine. Its contents define the Monte Carlo simulation specifications including input and output file names and directories, the number of scenarios to be generated, which parameters are to be randomized, and

Figure 4.1: Proposed evaluation framework, from data generation to analysis of results.

the probabilistic details of the uncertainty.

- **MCSim Simulation Engine:** When completed, this program will read in the simulation specification file and provide values for the randomized variables to the scenario builder.
- **MS-Epanet Input File (*.inp):** This input file to MS-Epanet specifies the network characteristics. A sample *.inp* file is included in Appendix B and

shows how the network layout and characteristics are defined, as well as how reaction models describing the fate and transport of the chemicals to be used are specified.

- MCSim / MS-Epanet Scenario Builder: When completed, this program will actually run the simulations using MS-Epanet. Each scenario, as specified by the MCSim Simulation Engine in accordance with the demand and attack specifications given in the **.sim* file, will be run using the network and water quality reaction models provided in the MS-Epanet input file.
- Scenario Ensemble Database: As the multiple scenarios are run, the results will be written to a **.sed* file. The format of this database has not yet been defined, but it will likely be developed in conjunction with utility personnel and data management specialists so that its format will be able to be used when CWS's are actually deployed.
- MS-EPANET Ensemble Execution Controller: This program, currently being developed, will manage the data generation phase by coordinating scenario runs (especially important if multiple processors are being used) and writing the data generated to the *.tso* file.
- Simulation Database File (**.tso*): The results of each simulation specified by MCSim and carried out using MS-Epanet, including concentrations of all specified chemicals at each node at every timestep, are written to a single output file. This is an unformatted binary file in order to save disk space

and computation time, and is not readable using an ordinary text editor.

The format of the file is described in Appendix B.

- TSO Formatter: Once completed, this program will extract the necessary data from the *.tso* file produced and rewrite it in the format used by the DLL (**.tao*). Note that the *.tso* file contains information that will not be used in this evaluation, including network details, hydraulic information, and data from non-sensor nodes. The efficiency of the algorithms and the time needed to complete the evaluation can be reduced by eliminating this unnecessary data and organizing the needed data in an easily accessible form.
- DLL Input File (**.tao*): The *.tao* file, read in by the DLL, contains all data needed to run the algorithm and complete the evaluation, including chlorine and contaminant data from all scenarios, sensors, and timesteps. The format of this file is described in Appendix B.
- *getData* DLL: The *getData* dynamically linked library (DLL) has been developed specifically for this evaluation as a means to interface with the different algorithms. The DLL takes the *.tao* file as input and sequentially feed the scenario data contained in it to the algorithms each time its main function is called. Also, it collects the status decision from the algorithm at each timestep, and writes this output to the *.out* file. The function characteristics ensure that the algorithms can only access the data in chronological order and cannot access future data when making decisions about the current state of the network. The DLL is described in more detail in Section

4.5.

- **Developers' Algorithms:** Each algorithm developer participating in this evaluation will be required to provide Windows executable(s) of their algorithm that interface as specified with the *getData* DLL. These programs will utilize the DLL but will run independently.
- **Output File (*.out):** The DLL writes the network status returned by each algorithm during each timestep to the *.out* file. The results written in this file will be analyzed, and this file can be inputted into the Analysis Tool for further investigation.
- **Output Analysis Tool:** Once developed, this program will take the algorithms' output (in the form of the *.out* files) and compare the results with the correct status information obtained from MS-Epanet. It will then perform statistical calculations for each algorithm's performance including detection times for each scenario and each algorithm's median detection times, false positive rates, and sensitivity and specificity.
- **Final Report/ Statistics (*.evo):** This file will be the output of the Analysis Tool and will contain statistical results about each algorithm's performance which will be used for reporting, comparing the algorithms, and constructing figures to visually represent the algorithms' performance. The format for this file has not yet been defined.

4.2 Solicitation of Event Algorithm Developers

A copy of the initial experimental plan was sent to 29 research groups developing algorithms that could be used for event detection in January 2006. A letter explaining this thesis project was also included which asked for their feedback and/or participation in the evaluation. The contacted groups represented a wide variety of specialties and included representatives from industry, university researchers, and government agencies. There was also a large variety in the analysis methods used by these groups, comprising various artificial intelligence (AI) and statistical techniques. Some of the algorithms were developed specifically for this problem of sensor data analysis in water distribution systems for event detection, while others were general algorithms designed to detect anomalies in any time series.

4.3 Finalize Evaluation Details

4.3.1 Feedback from Developers

Feedback was received from 13 of the contacted developers. All of those responding were positive about the project, and no one raised major objections or indicated any major flaws in the plan. The only comment that proposed a change was one group's request that flow data be provided along with the chlorine data. This request was denied on the basis that this is preliminary evaluation aimed to test the algorithms' ability to find irregular states in a time series without bringing in extra information such as hydraulics.

Five of those responding agreed to submit their algorithms to be evaluated, with the remaining eight declining because of time restrictions or their belief that their algorithm was not easily compatible with this testing situation. See section 4.4 for a summary of the participating algorithms.

4.3.2 Challenges

As the experimental framework was finalized and specific procedures for executing and scoring an evaluation were defined, several unforeseen challenges arose. Most were run-of-the-mill setbacks, such as file manipulation issues in the DLL. But some were more substantial and are likely to be encountered by future evaluators. These challenges are described below.

Definitions

The first major issue encountered, and one that still endures, was the problem of defining the terms false positive, true positive, false negative, and true negative. Since the algorithms are required to output a status decision at every *timestep* for this evaluation, the first idea was to classify each timestep, using the following definitions. The algorithms response at timestep t for sensor s is considered to be a

- false positive if $truestatus = 0 \wedge algorithmreturn = 1$
- true positive if $truestatus = algorithmreturn = 1$
- false negative if $truestatus = 1 \wedge algorithmreturn = 0$

- true negative if $truestatus = algorithmreturn = 0$

where 1 indicates that an attack is occurring, 0 indicates that conditions are normal, $truestatus$ is the actual state of the system as determined by the contaminant concentration at time t and sensor s outputted by MS-Epanet, and $algorithmreturn$ is the status decision indicated by the algorithm.

However, this definition did not seem to fairly represent all systems. For example, some systems are designed to only indicate the onset of an event, not to continue to alarm throughout the entire event. These types of algorithms would score very poorly even though they may detect the attack quickly. Assume that an event is n timesteps long, and that a particular algorithm detects it immediately (in the very first timestep it appears at the sensor). This definition would correctly credit the algorithm with a true positive in the first timestep, but would count the remaining $n - 1$ timesteps where it did not re-identify the attack as false negatives. This hardly seems fair. And even for an algorithm designed to signal for the entirety of the attack, if the algorithm continued to signal an attack a timestep or two after the event ended, these would be considered false positives. One suggested solution was to consider all timesteps from the first point of detection to the end of the event to be true positives, regardless of the status returned by the algorithm. Another was to put a one- or two- timestep buffer at the end of each event, not penalizing any attack signal given during those times.

The other approach considered was to classify each scenario / sensor combination instead of each timestep. The definitions for this method got a little fuzzy. It

was fairly easily decided that if *algorithmreturn* = 1 at a sensor for any timestep for which *truestatus* = 1 as well, then that scenario would be considered a true positive at that sensor. But if the event was not detected while contamination was present but sent an alarm 10 steps later, should this be considered a false negative since the event was not detected, or a false positive since there was an alarm sounded outside of the event period? And what if the algorithm just outputted 1's for every timestep? Would any scenario that contained an event then automatically be considered a true positive? All of those false positives, which are important to consider, would be lost in this scoring system.

This issue was discussed at length and never fully resolved. But it was decided that a combination of these approaches would be used for the evaluation. Details on the final performance measure definitions can be found in section 4.7.2.

Another issue was encountered when sample simulations were run using MS-Epanet. It was noted that a single contaminant injection showed up at each individual sensor location differently (see Appendix E). A particular injection attack may appear more than once at sensor *a* because of changing network hydraulics, but may never pass sensor *b*. This difficulty was fairly simply remedied, however, by defining two different terms: an attack and an event. An attack was defined as the period of time during which contaminant is actually injected into the system. For example, an hour long injection of sodium arsenite at sensor 32 at a rate of 60 gallons per hour would be a possible attack. An event was defined to be a continuous period of time when contaminant is present at a particular sensor. So if the attack described above caused a non-zero level of sodium arsenite at sensor

two from 8:00-8:30, then the level dropped back to zero but then rose again from 10:15-11:30, sensor two would be said to have encountered two events as a result of this one attack.

Expectations

This experimental proposal was presented to several groups during this development that included algorithm developers, researchers involved in other algorithm evaluations, chemists, and water distribution system employees. With each presentation, the feedback was positive, but several suggestions were given. These included incorporating real network data into the evaluation, including hydraulic data, and to consider how the water distribution system personnel would likely respond to an alert (for example, to not act on an alert unless it is repeated several times). With each presentation, the suggested scope of the work seemed to expand. And though these suggestions were interesting to hear and will likely be addressed in the future, it was necessary to save these ideas for the future work and stay focused on the original intent of this project.

4.3.3 June Workshop

As this work was presented, it became clear that many of the difficulties encountered were of interest to others in the field, specifically those working on the WaterSentinel project (see section 7.1.1). Therefore, I worked with some fellow researchers to schedule and plan a workshop on June 20 and 21, 2006 at the USEPA office in Cincinnati, OH. Approximately 25 people participated in the workshop

including water distribution system personnel, event detection algorithm developers, researchers involved in the evaluation of public health surveillance systems (a field very similar to the water quality problem), and researchers working on the ASSET (described in section 7.2.1) and WaterSentinel (section 7.1.1) programs.

In planning the workshop, the following issues / questions arose as key discussion points.

- Defining an event: What types of events should a CWS detect? Significant water quality changes or just those that may indicate contamination? If the contaminant passes by a sensor for two hours, is that considered one event, or 120 events (if the sensor sends data every minute)? What about if it goes past a sensor for an hour and then hydraulic changes cause it to come back two hours later: is that one or two possible detections?
- Defining the time period of the event: Is the event characterized by the total time that a contaminant passes by the sensor, or just the beginning or end of the event? Should the event time correspond to the time in which the contaminant is present, or when the water quality parameter is affected? How would that be determined? What if the contaminant is present at such a low level that there is no noticeable water quality effect? Should the algorithms be expected to detect this?
- Defining a detection: How can it be determined that this algorithm is actually detecting the event, and it is not a false positive? What is considered a successful detection? Should an algorithm identify an event for the entire

time period when a contaminant is present, or is the initial alarm sufficient? What if a CWS detects the event 15 minutes after the two-hour event has concluded? What if a CWS continues to detect the event for several timesteps after the event has ended?

- Identifying key performance measures: Which performance measures are most important? Is a high detection rate or low false alarm rate more important? How about detection time? Should performance be weighted based on the size or impact of the detected / missed attack? Should software performance measures such as efficiency and scalability be considered?
- Data needs: What type of data will be used? How much data is necessary for training and testing purposes? How realistic does the data need to be? How many events need to be simulated in order to develop reliable performance estimates? How should events be constructed (e.g., different contaminants, starting concentrations, injection durations, etc.) Will the data be perfect or flawed? In what format will the data be? Does the data accurately represent that which will be seen when the algorithm is placed in an actual network?

The first day of the workshop consisted primarily of presentations by the various participants. Presentation topics included a review of event detection algorithms in the public health surveillance field, descriptions of some anomaly detection algorithms, an overview of current water quality data collection efforts for use in algorithm analysis, and updates on current evaluation projects of water

quality event detection algorithms. This evaluation proposal was presented and received very favorable feedback.

The second day was focused on defining the terms and addressing the questions listed above. With the variety of people present with such a range of experiences, conversation was plentiful and viewpoints were diverse.

An event was the first term addressed, and even this seemingly straightforward term sparked hours of discussion. The first question raised was exactly what the algorithms should detect. If the system was only to be used to detect contaminations, a definition like the one used for this project (a non-zero level of contaminant at a sensor) would be appropriate. However, some argued that this definition was too narrow, and that the algorithms should detect any change in water quality. It seemed to be the general consensus that the detection of any water quality change would be beneficial, except for predictable ones caused by system changes such as pumps turning on or off or mains being flushed. Though some of the more mathematically-inclined participants would have preferred a more precise definition, it was decided that an event would be defined as an unexpected, noticeable change in a water quality parameter.

This definition generally pleased everyone, but opened the door for many more, mostly unanswered questions. What is a noticeable change in water quality? What is the use of an algorithm if it simply detects an already evident change? What if a contaminant is present at a sensor but there is no 'noticeable' water quality change? Is that not an event that should be detected? If non-contamination events are to be evaluated, how will the beginning and end of an event be defined,

as there is no measurable quantity (such as the level of contaminant) to indicate when an event is occurring? One interesting point brought out by those working in water distribution systems was the desire for the algorithms to signal for all timesteps during an attack, not just the beginning. They noted that multiple alarms would lend credibility to the threat and would increase the likelihood that action would be taken.

Much time was also devoted to the scoring procedure for the evaluation. It was pretty well agreed upon that a ROC curve (see section 4.7.3) is an important analysis tool. However, that is where the agreement ended and the problems began. Like was previously experienced in this project, the measures associated with a ROC curve including false positives, true positives, false negatives, and true negatives were not easily defined. Again, there was discussion about whether to score each timestep individually or to score each event as a whole. Disagreement occurred on whether or not it would be a false positive if an algorithm continued to indicate an attack for a timestep or two after the event ended. Some suggested that two false positives in consecutive timesteps be counted only once.

Participants with experience with ROC curves noted that traditionally, data with attacks present is used to measure the sensitivity, or proportion of the attacks detected, and that attacks are looked at as a whole. For example, if five out of 10 attacks used to test the algorithm were detected, the algorithm's sensitivity would be $1/2$. The specificity, or proportion of the non-event timesteps where the algorithm correctly identifies that conditions are normal, is measured using only non-event data (so actual data collected from water utility sensors could be

used) and by looking at each timestep separately. For example, if there are 100 timesteps of non-event conditions, and the algorithm signals an attack for two of those, the specificity would be 98/100.

A few other suggestions for performance measures were suggested in addition to the standard ROC curve and detection time measurements. One was to measure the cost benefit of each algorithm by considering the cost to buy and maintain it, the cost of investigating the number of false alarms produced by the algorithm, and the savings, both monetary and non-monetary (lives), obtained by the detection power of the algorithm. Another suggested was to look at the ratio of false positives to true positives. This ratio shows which fraction of alerts made by the algorithm are actually true positives, and the lower this number is, the more likely a quick response will occur. It was maintained that this gives a more realistic view of the algorithm's performance, as actual events are so rare, and the majority of the alarms will, in fact, be false positives.

Another question that arose when algorithm evaluation was being discussed was whether to look at each sensor individually or to look at the network as a whole. This evaluation plan asks the algorithms to look at each sensor separately and requires each sensor to submit its own decision on the state of its location for each timestep. But it was the general consensus that future evaluations should require the algorithms to output the status of the network as a whole. It was noted that, to date, no known detection algorithms synthesize data from different sensors to make a network-wide decision. In fact, to do so would require providing the algorithms with additional hydraulic information so that the relationship between

the sensors could be established (for example, if and when the water from sensor *a* will reach sensor *b*). Even so, current algorithms could easily use the outputs from the individual sensors to obtain a network status: the easiest solution would be to allege a network attack if any of the sensors signal an event. Also, it was noted that current algorithmic limitations should not dictate a simplistic evaluation. In fact, evaluation specifications such as this one could, in fact, inspire developers to improve their algorithms. The discussion of the many scoring issues was settled when it was pointed out that as long as scoring was consistent, the details were not all that important. If the algorithms are being scored in the same way, they will all fail if a particular objective is unattainable, and this will not affect their overall ranking.

The final topic was what kind, and how much, data is necessary for an acceptable evaluation to be run. The consensus was that real network data is ideal if available, as it provides a realistic test for the algorithms. However, it was noted that the volume of this historical data is limited and that it does not contain events which with to test the algorithms. Therefore, a combination of historical and quality simulated data (such as the data being used here) was suggested.

There were also no clear guidelines for the amount of data needed. The first, half serious, answer to that question was 'as much as the algorithms need.' Another was 'enough so that the patterns can be established,' so that at least a year of data would be necessary if yearly patterns were to be learned. It was noted that the more data available, the better. However, especially with historical data, the desired volume does not exist and adjustments need to be made to make the

most of what is available.

Overall, the workshop seemed to be interesting and helpful to the participants. Many issues were discussed, a variety of topics were presented, and numerous new questions were raised which will likely drive event detection algorithm analysis forward. One clear conclusion of the workshop was that an algorithm is only as good as the data that it receives. Therefore, work needs to be done to provide large amounts of a variety of data (various water quality measures, hydraulic information, and even non-system information such as 911 calls) in order for the most effective detection algorithms to be developed, tested, and improved.

4.3.4 Final Experimental Plan

No major changes were made to the experimental design, since the feedback received from the developers and at the workshop was positive. The only suggested improvement was to include more hydraulic and system information with the data. While this will additional information will certainly be a part of future studies, it was decided that incorporating hydraulics was beyond the scope of this initial work, which was simply intended to test the algorithm's ability to detect anomalies in a time series.

The only real modification to the experimental plan came with the distinction between an attack and an event. The original plan indicated that each scenario would contain at most one attack, and it was assumed that each sensor's time series would also contain just one attack. However, due to changing network

hydraulics, it was noted that one attack can show up at a particular sensor in several separate time period. Therefore, while it remained true that there would be at most one network *attack* during each scenario, the final plan allowed for multiple events to occur in a single time series.

The final experimental plan also included more details than the original design. The interface between the algorithm and the DLL was more explicitly defined, the actual performance measures to be used were listed, and the guidelines for training the algorithms and submitting the executables was described. The final plan is described in the letter to be sent to the algorithm developers along with the training data (as soon as it is available) and is included in Appendix D. Note that, once received, each group will have one month to modify and train their algorithms to comply with the evaluation requirements.

4.4 Algorithms to be Evaluated

Of the 29 algorithm developers contacted, five agreed to submit their algorithms for evaluation. Quite pleasantly, the algorithms encompass a wide variety of detection methods, and the algorithm developers come from a variety of settings as well. Each is described briefly here, and references are included if more details are desired.

- Change Detection Methods

Sean McKenna at the Sandia National Laboratory has developed two algorithms based on statistical methods[McKenna et al., 2005]. One uses linear

filtering and other uses lag-one differences, but both have been developed for this specific problem of event detection in a water distribution system.

- NormNet

NormNet [nor, 2006] is an "abnormality detection" algorithm originally developed by Frontier Technologies to sense when a jet engine was about to fail. This algorithm, which uses AI techniques, has since been adapted to other problems.

- Windowed Feature Extraction

Ahmet Bulut leads a group at the University of California, Santa Barbara that has developed an algorithm for analyzing data streams. It is based on a wavelet transform and uses statistical methods to detect abnormal conditions[Bulut et al., 2005].

- ADWICE

The Anomaly Detection with Real-Time Incremental Clustering (ADWICE) tool has been developed by Simin Nadjm-Tehrani at the University of Linköping [Burbeck and Nadjm-Tehrani, 2004]. This algorithm uses an existing clustering framework to implement fast, scalable, and adaptive anomaly detection. It has primarily been tested on IP network intrusion problems.

- Wavelet methods

Galit Schmueli and her students at the University of Maryland will submit an algorithm for evaluation based on wavelet methods. It is also a general

anomaly detection algorithm [Schmueli, 2005].

4.5 Implement Test Bed

The test bed will be executed through the use of a Windows DLL file that contains the function

$$\text{floatgetData}(\text{intstatus}, \text{int} * \text{timestep}, \text{int} * \text{sensor}, \text{int} * \text{truestatus}) \quad (4.1)$$

This function will be used by the algorithms to obtain the necessary sensor information, both for training and for the final evaluation, though the *truestatus* variable will only be enabled for training. *getData* returns the chlorine sensor reading for the given timestep from the indicated sensor, and a value of -1 is returned at the final call to indicate that the data from all sensor and scenarios has been processed. It is designed to provide the algorithm with sensor data, stepping consecutively through the time series from each sensor and scenario.

**timestep* gives the timestep of the data currently being returned. A **timestep* value of 0 indicates that a new time series / scenario is beginning. **sensor* tells which of the sensors (0-4) is being read during the current time series. **truestatus* (0 or 1) tells whether there is actually contaminant present at the current sensor and timestep. Note that this parameter will only be activated for testing, so that the algorithms can determine what the correct output should be for each timestep.

status is an input parameter that must be that provided by each algorithm at each timestep in order to receive the data for the next timestep. It indicates the algorithm's determination of the status of the network during the previous timestep. This value will be 0 if the algorithm decides that no contaminant was present in the network, 1 if the algorithm deems that contaminant was present, or -1 for the very first call to this function. A valid value must be provided in order to receive the next simulation value, and these status reports are saved to a file that cannot be accessed or changed by the algorithms.

A simple program that uses the executable is shown in Figure 4.5. Note that *status* is set to -1 the first time *getData* is called. The *while* loop checks to see when all data has been processed, as indicated by a -1 being returned. And in the loop, this program cheats and simply returns the status it knows is correct by checking the *truestatus*. It would not be able to do this during the actual testing, as true status will be disabled for the final evaluation.

All participating groups will receive a Windows DLL file containing the *getData* function needed to access the training data and a detailed description of the instructions and final experimental design (see Appendix D). The training data will consist of 250 separate time series, each consisting of chlorine data from one of the five sensors from one of the 50 one-week simulations. Each run will be generated to represent uncertainty and variability in various processes affecting real distribution system performance. There will be no missing data. All simulations will begin at the same time of day.

Developers will be asked to provide three versions of their algorithm. The three

c:\Documents and Settings\Katie Umberg\My Documents\Thesis\dllTester\main.cpp

1

```
#include <stdlib.h>
#include <stdio.h>
#include "getData.h"

void main()
{
    float current;
    int timestep;
    int sensor;
    int truestatus;
    int i=1;

    current = getData(-1, &timestep, &sensor, &truestatus);

    while(current != -1)
    {
        if ((truestatus==0)){
            current = getData(0, &timestep, &sensor, &truestatus);
        }
        else{
            current=getData(1, &timestep, &sensor, &truestatus);
        }
    }
}
```

versions will correspond to different target false positive rates. One algorithm should average a false positive no more than once every day, one should have no more than one false positive per week, and one should seek to reduce false positives to one per month.

The three versions of the programs will be submitted as Windows executable files. The executable programs will call the function *getData* as described above in order to get the chlorine data to analyze as well as the timestep and sensor information, and the programs must output the status for each timestep. There will be no time limit for analyzing the data and outputting the 0/1 status value, but the total running time for each algorithm will be noted in order to provide some measure, though inexact, of the relative efficiency of the algorithms.

The algorithms will be tested using both one-week contamination and non-contamination scenarios, reflecting the same statistical assumptions made to gen-

erate the training data. All algorithms will be tested using the same dataset. The series of 0's and 1's output by each algorithm will be collected and analyzed.

4.6 Develop Simulated Data Streams

The general framework was established in the original evaluation plan (see section 4.1.4) for how an innovative, realistic dataset would be generated. As the experimental plan progressed, scenario details such as the network, sensor locations, contaminants, and attack type were decided on and are described here.

4.6.1 The Network

A small midwestern utility was chosen with which to generate the simulations. It is an actual water distribution system, and a fairly sophisticated EPANET model has been developed and calibrated for it, including a 16-day demand pattern model. It is a fairly simple system, with few loops or unusual hydraulic conditions. The EPANET network model includes 482 nodes or junctions, 538 pipes, no reservoirs, six tanks, four pumps, and nine valves. The layout of the network is shown in Figure 4.2, with the stars indicating sensor locations. Note that the research teams were given no information about the network or sensor locations before they trained the algorithms. This ensured that they could not use spatial or network information, as this study seeks to measure their ability to detect an attack solely from a time series of chlorine information.

Figure 4.2: Network map with five sensor locations indicated

4.6.2 Sensor Locations

Once the network was chosen, five sensors (an arbitrary number) were placed in the network using the SPGreedy algorithm described in chapter 3.5.2. The algorithm uses a ranking procedure that attempts to minimize the expected health impacts associated with intentional distribution system contamination, assuming perfect contaminant-specific sensors[Uber et al., 2004]. This ranking procedure is in general non-optimal, but does account for uncertainty in the injection location as well as network fate and transport and estimated health effects. Thus these

locations are valid CWS sensor locations for the test network, consistent with current design procedures. Again, these locations can be seen on the network map in Appendix 4.2.

Figure 4.3: Water Age Signatures: A one-week time series of water age was created for each of the 5 sensor locations.

To ensure that these five sensors would provide a varied and interesting challenge, the locations were considered more carefully. First, a one-week water age simulation was run under non-event conditions and deterministic source and wa-

ter demand characteristics; results are shown in Figure 4.3, and show that the locations exhibit a variety of water age signatures. Since free chlorine residual dynamics are related to water age dynamics, the five locations should also represent a range of background free chlorine dynamics.

To get a better idea of how an attack may show up at the different sensor locations, two completely random attacks were simulated at two different locations in EPANET using an arbitrary contaminant tracer. The first attack was six hours long, beginning six hours into the study, and had an injection rate of 30,000 gallons per minute. The other attack was lasted for only two hours but had a higher injection rate of 50,000 gallons per minute. It began 12 hours into the simulation. Again, these attacks had very different impacts on the different sensors, including the fact that attack two was never seen by sensor 2 or sensor 4. The locations of the attacks and the time series of the contaminant levels at each sensor can be seen in Appendix E.

4.6.3 Contaminants

Sodium arsenite and a pro-oxonic organo-phosphate (OP) were selected as the two contaminants used to test the free chlorine sensor CWS performance. They were chosen because their reaction dynamics are known through literature and experimental testing, and they react very differently with chlorine. Therefore, they will provide very different challenges to the algorithm. Both have also been identified as potential threats because of their toxicity and availability.

Sodium arsenite is a soluble inorganic (As[III]) that reacts with chlorine to produce sodium arsenate, a soluble inorganic (As[V]) that is much less toxic than the arsenite. The half-life of the reaction with free chlorine is about 15 seconds (i.e. , which is virtually instantaneous considering our sensor timestep of 15 minutes). This will create a sharp drop in chlorine levels in the system and represents perhaps one of the most optimistic cases for the use of free chlorine residual sensors to detect a contamination event. The equations describing the reactions between sodium arsenite, sodium arsenate, and chlorine are

$$\frac{d[Cl_2]}{dt} = -k_b[Cl_2] - k_2[Cl_2][AsIII] \quad (4.2)$$

$$\frac{d[AsIII]}{dt} = -k_2[Cl_2][AsIII] \quad (4.3)$$

$$\frac{d[AsV]}{dt} = k_2[Cl_2][AsIII] \quad (4.4)$$

where $k_b = 6.9444E - 4$ is the daily free chlorine bulk decay constant and $k_2 = 69.4$ is the arsenite oxidation velocity [Dodd et al., 2006]. Figure 4.4 shows the interaction between chlorine and sodium arsenite.

A Pro-oxonic OP is a toxic organophosphate that creates an even more dangerous toxic oxon (claimed to be 5-1000 times more toxic than the parent) when it reacts with chlorine. The reaction is highly dependent on the pH of the water, but the half-life of the reaction from pro-oxonic OP to the oxon is about a half

Figure 4.4: Plot showing the interaction between chlorine and sodium arsenite.

hour, and the half-life of oxon is approximately eight hours. These times are much longer than the 15 seconds of sodium arsenite, and the change in chlorine could be more gradual and drawn out over several timesteps. The following reaction equations show the dynamics between the two compounds and chlorine.

$$\frac{d[Cl_2]}{dt} = -k_b[Cl_2] - k_r[Cl_2][OP] \quad (4.5)$$

$$\frac{d[OP]}{dt} = -k_r[Cl_2][OP] \quad (4.6)$$

$$\frac{d[Oxon]}{dt} = k_r[Cl_2][OP] - k_p[Oxon] \quad (4.7)$$

where $k_b = 6.9444E - 4$ is the daily free chlorine bulk decay constant, $k_r = 1.89$ is the OP reaction rate, and $k_p = 6.9444E - 4$ is the oxon reaction rate [Zhang and Pehkonen, 1999]. Figure 4.5 shows the interaction between chlorine and OP under two pH conditions. Note the spike in the oxon byproduct after the OP injection.

Figure 4.5: Plot showing the interaction between chlorine and OP under two pH conditions.

4.6.4 Attacks

Of the 100 scenarios run on MS-Epanet (50 for training and 50 for testing), 20 will be run without any attack on the network (10 for training and 10 for testing). These will be used by the developers to train the algorithms for the natural variation in the network without a contaminant, and in the testing they will provide a test to ensure that the algorithms are not detecting attacks that did not

occur. The other 40 scenarios in each set will include one attack with random characteristics as described in chapter 4.1.4. Note that not all attacks reach all five sensors, and some attacks may show up at a particular sensor as more than one event, so each individual sensor will actually likely not see exactly 40 events: the number may be higher or lower depending on the sensor location and attack locations used in the evaluation.

4.7 Scoring Procedure

An output file of 0's and 1's, indicating the event status, will be produced when each algorithm is run on the test data. These outputs will be compared to the correct values, which will be obtained from the simulated contaminant levels outputted by MS-Epanet, to evaluate the algorithms' performance. The following specific measures will definitely be calculated using these results, though additional data analysis will likely be performed once the evaluation begins.

4.7.1 Definition of an Event

As previously mentioned, two different terms require definition for this evaluation. First, an *attack* is considered to be the action of actually injecting a contaminant into the potable water supply, and is defined in terms of particular characteristics of location, start time, duration, and mass rate. For an example, one possible attack is a 3-hour attack injection of sodium arsenite at node 25, beginning 3 hours into the simulation and having an particular mass injection rate of 1000

gallons per minute. Second, an *event* occurs in response to an attack, and is defined as n consecutive timesteps at a particular sensor location during which the contaminant level is greater than a specified threshold (nominally, zero). Note that in contrast to an attack, an event is largely dependent on network hydraulics. For a particular scenario (during which at most one attack occurs), in general some sensors will not observe any events, whereas others may observe one or more events.

Figure 4.6 shows a hypothetical contaminant time series at a sensor location. (Note that there are no units on the y-axis, as they are unnecessary here; all that is important is whether the contaminant level is above or below a threshold, taken here to be zero). Two events occur during the time series shown. One begins at timestep two and continues through timestep six. The other event lasts from timesteps 17 through 19 (note that the first timestep is counted as 0).

Figure 4.6: Sample contaminant levels obtained from a sensor during a 21 time-step period.

Ideally, a perfect algorithm would output the following status series if provided

this hypothetical time series:

0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0

These would be the true status values used to evaluate algorithms performance.

4.7.2 Detection Objectives Measured

A variety of performance measures will be used to evaluate and compare the algorithms. Totals will be calculated for the entire test scenario ensemble, and for certain ensemble subsets that classify scenarios by contaminant type, event duration, and injection rate.

- True positives: Time steps during which an event is occurring and the algorithm correctly identifies that the event is occurring. For example, in the time series from figure 4.6, it would be considered a true positive if an algorithm produces a 1 for timestep four. Also, since it was decided for this evaluation that the initial event alarm from an algorithm is most important as it initializes the response, once event e has been detected with a true positive, all subsequent timesteps in event e , or until a 0 occurs in *truestatus*, are considered to be true positives as well. So if the algorithm outputted that 1 during timestep four, timesteps five and six would also be considered true positives, regardless of the status outputted.
- False positives: Time steps during which there was not an event occurring but the algorithm indicates that contaminant is present. In the example in figure 4.6, an output of 1 in the first timestep would be a false positive.

- True negatives: Time steps for which the algorithm correctly identifies that there is no event occurring. In the previous example, it would be considered a true negative if an algorithm outputted a 0 in the first timestep. Note that the sum of false positives and true negatives will be equal to the number of timesteps with a true status of 0.
- False negatives: Time steps during which an event is occurring, the event has not yet been detected, and the algorithm outputs that there is no event. If the algorithm outputted a 0 for timesteps two, three, and four of the example indicating that conditions are normal, these would be false negatives. Note that a 0 in timestep four would be a true positive, however, if the algorithm had outputted a 1 in either timestep two or three (see the definition of a true positive). The number of true positives and false negatives will add up to the total number of timesteps during which an event is occurring.
- Time to detection: Time between the beginning of an event and the first 1 outputted by the algorithm during the event. If, in the example shown in figure 4.6, the algorithm produces a 1 timestep two, the detection time would be zero. If it outputs a 0 in the second step but a 1 in the third, the detection time would be one timestep, or one times the sampling interval. When using time to detection to compare the algorithms, the median value will be used, with the detection time being infinity if a 1 is not outputted by the end of the event.
- n -step Detections: The number of events which the algorithm detected

within n time-steps. For example, if an algorithm took 3 timesteps to detect event a and the detection time for event b was 8 steps, the algorithm would have no 2-step detections, one 5-step detection, and two 10-step detections.

The metrics listed above are fairly standard, but their values are critically important for measuring algorithm performance. True positives and true negatives are obviously desirable, as the goal of these algorithms is to determine the current state of the network. False negatives allow an event to occur undetected, and false positives cause unnecessary worry and confirmation costs.

The time an algorithm takes to detect an intrusion is also extremely important, as the contaminant will continue to move through the system and more of the population will be affected as time progresses. These algorithms will ultimately be run in real-time, and the faster that the algorithms can analyze the data and detect a problem, the better. Especially since the follow-up actions including confirmatory sampling, public address alerts, and system shutdown are time consuming, a timely alarm is almost as important as a correct one.

Additional metrics will likely be calculated using the binary series obtained by the algorithm output. For example, the actual output streams could be evaluated to see if there are any statistical patterns in the algorithm outputs, such as predictable runs of 1's and 0's. Also, the number of contamination scenarios detected by multiple sensors could be calculated.

4.7.3 ROC curves

Once the algorithms have been run and the 0/1 status reports have been collected, each timestep will be classified as a false positive, false negative, true positive, or true negative. In addition to the raw values of these metrics, the tradeoffs between them (e.g., the tradeoff between true positive rate and false positive rate) is fundamental for evaluating sensing systems. Using these totals for false positives, false negatives, true positives, and true negatives for each of the three threshold settings, a receiver operating characteristic (ROC) curve will be approximated for each algorithm. A ROC curve is a 2-dimensional plot of sensitivity versus specificity. In this case, sensitivity is defined as the ability of an algorithm to correctly classify a network as contaminated, or the number of true positives divided by the total number of timesteps during which contaminant is present at the sensor (equation 4.8). Specificity is defined as the ability to correctly classify a network as normal, or the number of true negatives divided by the total number of timesteps during which there was no contaminant present (equation 4.9).

$$sensitivity = \frac{\sum_{i=0, nscenarios} \sum_{j=0, nnodes} \sum_{k=0, nsteps} tp(i, j, k)}{\sum_{i=0, nscenarios} \sum_{j=0, nnodes} \sum_{k=0, nsteps} actual(i, j, k)} \quad (4.8)$$

$$specificity = \frac{\sum_{i=0, nscenarios} \sum_{j=0, nnodes} \sum_{k=0, nsteps} tn(i, j, k)}{\sum_{i=0, nscenarios} \sum_{j=0, nnodes} \sum_{k=0, nsteps} 1 - actual(i, j, k)} \quad (4.9)$$

where *nscenarios* is the total number of scenarios that were used to test the algorithms, *nnodes* is the number of nodes in the network used in the evaluation, *nsteps* is the number of timesteps for each scenario, *tp*(*i, j, k*) is 1 if timestep *k* for

sensor j in scenario i is a true positive (there is a contaminant present there at the given time and the algorithm correctly returns this) and 0 otherwise, $actual(i, j, k)$ is 1 if there is contaminant at sensor j during timestep k in scenario i and 0 if no contaminant is present, and $tn(i, j, k)$ is 1 if timestep k for sensor j and scenario i is a true negative (the algorithm correctly determines that no contaminant is present).

A point is plotted for each of the three algorithm versions provided (corresponding to different threshold settings), with the x-coordinate being 1 - specificity, or false alarm rate, of the version and the y-coordinate representing the sensitivity, or detection rate. A curve will then be fit through the points, and this resulting curve can give information about the performance of each algorithm. Although only three points will be available for this study with which to approximate each ROC curve, at least a rough ROC curve will be drawn to provide a general feeling for the performance characteristics of the different algorithms.

In general, most ROC curves will go through two points: $(0, 0)$, and $(1, 1)$. The point $(0, 0)$ will occur when the thresholds are set to an extremely high level. In this case, there will be no detections or false alarms, as nothing will be ever be detected. The other point $(1, 1)$, is produced when thresholds are set to such a low level that all events are considered to be attacks. In this case, the detection and false alarm rates would both be 1, as an alarm will be made for every timestep. The only curves that would not go through these points would be produced by a perfect algorithm, which would consist of only the point $(0, 1)$. This would indicate that all of the attacks are detected by the algorithm regardless of the

threshold settings (sensitivity = 1), and there are no false positives ($1 - \text{specificity} = 0$). An algorithm that simply guesses the state of the network randomly would produce an ROC curve that is simply a straight, 45 line, as there would be an equal probability of guessing correctly or incorrectly at each timestep. An effective algorithm will lie above this 45 line, as it should perform better than a random, uninformed algorithm.

Figure 4.7: A sample ROC Curve. The dashed line indicates the worst possible performance of a CWS in which no useful information is provided. The farther the curve is to the top right of the graph, the better the algorithm performance.

A quick look at the shape of the curves produced by different algorithms can give a preliminary comparison of their effectiveness: the faster the detection rate (sensitivity) increases, or the closer the curve hugs the y-axis, the better the algorithm. Poorer algorithms would have curves that follow the 45° line more closely. A comprehensive scalar measure of algorithm performance is obtained as the area under the different ROC curves. "The greater the area under the curve, the better the accuracy. Tests with an area under the curve of 0.5-0.7 have low accuracy, 0.7-0.9 moderate accuracy, and ≥ 0.9 high accuracy. An ideal test has

an area under the ROC curve of 1” (Collinson, 1998).

A ROC curve shows the tradeoff between detection rate and false alarm rate, so in addition to helping to compare the overall performance of the algorithms, CWS developers can use ROC curves to make an informed decision about which threshold values to choose based on their specific detection and alarm needs. The shape of the curve can also be used to choose threshold levels. ”The instantaneous slope of the curve demonstrates the rate at which the sensitivity changes with changes in specificity” (Morrison et al, 2003), and it has been stated that ”where the point of maximum curvature occurs corresponds to the optimal trade off between sensitivity and specificity, and this is the optimal cut off value for the test” (Collinson, 1998).

ROC curves were first used during World War II to analyze radar images and distinguish between an enemy ship, a friendly ship, and noise. They are currently used extensively in the biomedical field to determine the effectiveness of diagnostic tests by considering their ability to distinguish between healthy and unhealthy patients (Morrison, 2003). However, ROC curves can be used to compare algorithms and determine thresholds in any problem setting ”that can be divided into a binary classification of the characteristic under investigation” (Collinson, 1998), such as the event detection problem where each timestep can be classified as either normal or attack conditions.

4.7.4 DET curves

Martin et al. note that "detection tasks can be viewed as involving a tradeoff between two error types: missed detections and false alarms" [Martin et al., 1997]. To illustrate this performance, they use a Detection Error Tradeoff (DET). This is similar to a ROC curve in that it describes an algorithm's ability to correctly identify events, though instead of plotting the percentage of correct detections on the y-axis, the miss probability is used. The false alarm probability is still on the x-axis.

By plotting error rates on both axes, it gives "uniform treatment to both types of error, and us[es] a scale for both axes which spreads out the plot and better distinguishes different well performing systems" [Martin et al., 1997]. Instead of curves crowding into the top left-hand corner like with ROC curves, DET plots produce more linear curves, with the line $y = -x$ indicating completely random performance. Figure 4.8 gives an example of how the plots differ.

4.7.5 Efficiency

Efficiency of time and memory will be extremely important when these algorithms are run on larger networks with more sensors and multiple water quality parameters. As this is simply an initial evaluation and the final real-time test bed is not yet in place, the efficiency will not be emphasized or precisely calculated. However, the total CPU time required by each algorithm will be noted. This will give some idea of how the algorithms compare in terms of efficiency, since they will all

Figure 4.8: The same speech data plotted using a DET curve (left) and a ROC curve (right). Taken from [Martin et al., 1997].

be run under identical conditions: on the same computer when no other programs are running on the computer and using the same data.

Chapter 5

Results

As is the case in so many evaluations, lack of data has held up the implementation of the experimental plan described here. This thesis work was begun with the assumption that the necessary dataset would be generated using the tool described in section 4.1.4 in early 2006, which would have allowed sufficient time for the completion of the project: algorithm developers would have had approximately one month to train their algorithms and send back executable(s), the executable(s) would have been run using the test data, and the results would have been summarized and examined.

The described data generation tool was not completed in time, however, and so the final execution of the evaluation was unable to begin. The project will be completed by another researcher once the data is available and results will be published when they are available. However, the lessons learned by working on the literature review, designing the experimental testbed, and planning and participating in the algorithm evaluation workshop (see section 4.3.3) will be valuable

for guiding future evaluations.

Chapter 6

Discussion

The experimental framework described in this thesis seems to be substantial, sound, and promising. Each of the numerous times that the proposed plan has been presented, the feedback received from the algorithm developers, participants of the evaluation workshop, and USEPA employees has been very favorable. Also, a paper describing this work has been accepted for the Water Distribution System Analysis (WDSA) Conference to be held in Cincinnati, OH in August 2006. This proposal is by no means a definitive experimental strategy. It has several weaknesses, including the fact that the data is not necessarily fed to the algorithms in real-time. However, as noted before, this is a preliminary work, and it can hopefully be used to inspire better, more realistic future evaluations.

6.1 Experimental Framework

One of the major objectives and motivations for this project was the development of a framework for testing event detection algorithms that can be used by future researchers, and the generation of suggestions for future work on event detection evaluations. The following guidelines are intended for anyone seeking to evaluate or compare event detection algorithms. The suggestions come from lessons learned while performing the literature review for this project, discussing the project with colleagues, and addressing problems encountered during the experimental design and implementation.

6.1.1 Suggested Steps

The following is a suggested approach to planning and carrying out a successful algorithm evaluation. Note that these are guidelines for the ideal evaluation, and it will not always be possible to address them all. In fact, several of these suggestions were not followed in this initial evaluation design, as it was a first attempt at this type of analysis. However, many of these ideas will be incorporated in subsequent studies.

1. Research the Problem Status

The first step in planning an evaluation is to determine exactly what needs to be evaluated. Therefore, it is helpful to research what has been done in the field, and what questions remain unanswered. It is foolish to spend the time and resources to redo what has already been done. Find out what still

needs to be examined, and tailor the evaluation to test that area.

2. Contact Stakeholders

The key to an informative evaluation is to simulate real-world conditions as closely as possible. Therefore, it would be helpful to communicate with those who know what the conditions are like. Contact the water distribution system where the algorithm may be used and establish a working relationship with the personnel there. Possible questions to ask a utility include:

- What historical water quality data do they have available?
- What format is their data in? How is it stored?
- What kind (if any) of sensors do they have in the system? What water quality parameters are currently measured?
- Do existing sensors collect data continuously, or how often do they collect data?
- What kind of water quality monitoring is currently in place?
- What is the protocol for dealing with alarms this system may raise?
- How are alarms actually dealt with in practice?
- What are they looking for in a CWS? Would they like it to detect any water quality abnormality, or just those due to deliberate contamination events?
- Would an initial alarm of an event be sufficient, or would they prefer the system to continue to alert throughout an event?

- How important are false positives? What would be an acceptable number of false positives?
- Is there any money budgeted for the installation and maintenance of a CWS? If so, what size system are they expecting to install?

The cooperation of the individual utilities will vary depending on their interest in CWS's. However, if a utility is willing to help, they can provide invaluable information, and possibly even some historical water quality data. Many utilities already have a limited number of water quality sensors in the system that are monitored, often to regulate chlorine levels.

Though a utility partner would certainly enhance an algorithm evaluation and add to its credibility by providing first-hand knowledge of actual network conditions and operating procedures, supplying actual water quality data, and giving insight into what is expected from a CWS, an evaluation can still be successful even if an enthusiastic utility cannot be found. From the utilities contacted as part of the current evaluation effort, it seems that most have some interest in a CWS, though the budget allotted for installation and running costs will allow for only a few sensors and limited personnel support. The utilities seem to prefer if the alarm continues throughout an attack, to really get the operators' attention and ensure a quick response. False positives are undesirable, and more than one or two a day can cause those monitoring the system to become frustrated and give less attention to those alerts that occur. Finally, SCADA is the most common format for

collecting and saving information from water quality sensors, and it is likely that the algorithms of a CWS will have to deal with SCADA data.

It would also be beneficial to contact any other stakeholders in the system and get them involved from the beginning. These may include politicians, researchers in the field, sensor developers, and algorithm developers.

3. Define Key Terms

Once the background research is complete, it is important to define some of the key (and often problematic) terms that are central to any evaluation including an event, a detection, and a false positive. This can be done in collaboration with the stakeholders and should reflect the goals of the evaluation. Some questions to ask include:

- Is an event simply the presence of a contaminant, or is any water quality abnormality considered an event?
- How will an event be defined? How will the beginning and end of the event be determined? (especially important if historical data is to be used)
- What will be characterized as a successful detection?
- Should the algorithm continue to alert throughout the attack, or is the initial alert sufficient?
- Will each sensor be considered separately, or will the algorithms be asked to provide a system-wide status?

- How will false positives be counted?
- Will there be any grace period at the end of an attack where false positives are allowed?
- Will factors such as the systems' efficiency or costs be considered, or is detection ability the main focus?

4. Define the Evaluation Specifications and Get Feedback

In the previous step, the basic goals of the evaluation are identified. Once this initial planning is complete and the terms are defined to everyone's liking, the details of the experiment should be established. This should be done without any input from those with a stake in the results to ensure an unbiased evaluation. The more detailed the evaluation description is at this point, the better, as many future headaches can be avoided by thinking through possible difficulties at the beginning.

Issues to address include, but are not limited to:

- What type of data will be used for the evaluation? Where will it come from? How much of it is available at this time?
- Is the data realistic and representative of data that would be seen in a distribution system?
- How will attacks be simulated in the data? Are these representative of the ones that are believed to be a threat to the system?
- What types of attacks will be simulated? What attack characteristics

(such as duration, start time, and injection rate) will be varied? Will attacks occur individually or could there be multiple attacks occurring at the same time? Is there a minimum wait time between attacks?

- Will the algorithm be tested on attacks similar to ones shown in the training data, or will new attack types be used to really test an algorithm's performance ability?
- Will the data be perfect or imperfect? Will sensor failure and transmission errors be considered?
- How will the algorithm developers train their algorithms?
- Several different settings will be needed for each algorithm in order to get the points to construct a ROC and to see some of the algorithms' possible functioning conditions. How will the settings, such as thresholds, be varied? Will the algorithm developers set them for target false alarm rates before they submit their algorithms? Or will it be required that the different parameters be adjustable by the evaluators?
- What format will the data be input to the algorithms? How will the algorithms access the data?
- What groups will be asked to participate in the evaluation? How will they be contacted?
- How will the algorithms be tested? Will the developers test their own algorithms using data provided? Or will they be asked to provide their algorithms for the evaluators to execute? What format should those

algorithms be in?

- What computer(s) will the algorithms be run on? What operating system will be used?
- What type of output is expected from the algorithms? A status report every timestep? A signal when an attack is suspected?
- Is a normal / abnormal indication by the algorithms sufficient? Or is a confidence level in the output desired?
- Is any additional information desired with an alert, such as an indication of what contaminant is being seen or where the attack is occurring?
- What performance measures will be used? Are measures such as flexibility and ease of use going to be considered? How about efficiency and memory usage? Will cost be considered?
- What will the final report look like? What figures will be included? What calculations will be performed?

Once the experimental design is complete, it can and should be shared with colleagues, stakeholders, and those who may participate in the evaluation. Their feedback can be helpful in identifying potential problem areas and areas that need more clarification. For example, algorithm developers can indicate if the experimental requirements are feasible and if the amount of training data proposed is sufficient. Be explicit in the description of the plan: define exactly how the evaluation will be executed and scored, and define any terms used. This is lacking in many experimental reports, and

the results are worthless without details: the experiment is not reproducible and it is often unclear what is even being measured.

One note about this, however: be sure not to provide the algorithm developers with any information that may influence the results. For example, in this study, the layout and hydraulic details of the network being used was kept from the participants until after the evaluation was complete so that they could not attempt to extract extra information for use during the evaluation.

5. Finalize the Plan and Gather All Necessary Materials

Based on the feedback received regarding the draft framework, finalize the evaluation details. It would also be beneficial to assemble all the necessary tools to carry out the evaluation at this time. This includes actually acquiring and preparing the data for input into the algorithms, developing and debugging any necessary software for interfacing with the algorithms or analyzing the results, setting up the hardware that will be used, and establishing which algorithms will be tested. It is advisable to thoroughly test the entire evaluation system, perhaps by completing the proposed evaluation on a test algorithm. This is another way to detect any potential problems early (for example, difficulty with software interactions with the algorithms), before the evaluation begins. It will ensure that the experimental framework is complete and achievable, facilitate the smooth execution of the evaluation, and add to the ease and professionalism of the study.

6. Carry Out the Evaluation After all of the effort put into the preparation of the experiment, the actual evaluation implementation should be straightforward. Once the testbed is ready, send the algorithm developers the final experimental plan and the training dataset, assemble the algorithms from the different research teams after the prescribed period of time, test the algorithms using the developed testbed, and collect the algorithm outputs. Note that the details of the evaluation, including what performance measures will be used, should be provided to the participants before they are asked to submit their algorithm. This ensures a fair and accurate evaluation, as all algorithm developers then tune their algorithms to the same criteria, and presumably the very best versions of the algorithms for the evaluation criteria are provided.

7. Report on the Results

Bring the evaluation to a close by reporting the results. The actual report should not contain opinions, but should factually state the outcome of the experiment.

Some possible questions to answer about each algorithm include:

- What was the algorithm's detection rate?
- What was the median detection time?
- What was the false alarm rate?
- What kinds of attacks was the algorithm able to identify correctly?

- What types of attacks were missed by the algorithm?
- Was the algorithm able to detect new types of data or attacks?

Sensitivity, specificity, and detection time are performance measures that are standard for this type of evaluation, though more will likely be included depending on the goals of the evaluation. Some clear tables and figures are helpful to illustrate and support the conclusions, including ROC curves and histograms. These tables and figures could show both the algorithms' performance on the testing ensemble as a whole, and the effectiveness on subsets of the scenarios used. For example, the attacks could be divided based on contaminant type, injection rate, or duration. It is valuable to construct a wide variety of figures of different types, showing different measures, and illustrating different attack set partitions. Non-obvious patterns can be uncovered in this way, and the figures that prove to be illustrative of algorithm performance can be included in the final report.

All results should be reported, whether positive or negative. Unless the participants agree to have their identities revealed, the participants should not be mentioned by name in the final report. Also, the participants should be allowed to publish their results, though again they should not mention other participants or their results.

It is important to note in any report that the results given only reflect the algorithms' performance on the given data and evaluation protocol. The results obtained should be viewed as a 'snapshot' of the algorithms' abilities,

and should not be construed as a definitive assessment.

8. Discuss the Results

Finally, the discussion section at the end of the report is the time to include any opinions, to discuss any oddities or unexpected results, and to draw conclusions. Depending on the results and what information was gathered on the systems, it may be appropriate to comment on the pros and cons of each system (such as ease of use, ability to detect a wide range of attacks, or efficiency). It is also appropriate to suggest future work, including possible improvements to the algorithms and to the experimental framework.

6.1.2 General Suggestions

In addition to the guidelines listed above, the following advice may be helpful to those seeking to carry out an algorithm evaluation.

- **Mimic, as Accurately as Possible, the Real-world Conditions the CWS will Encounter**

The truth of the matter is, all that really matters is how the CWS performs when it is deployed in an actual water distribution system. So no matter how elegant or 'insightful' an evaluation is, it is of little use if it does not accurately represent the true behavior of the system. The testbed should be developed in order to create the most realistic environment possible, from the input format and delivery, to the data itself, to the output required.

The experimental plan described in this work is focused on testing algorithms using realistic data, and the data used will be the most realistic simulated data yet used for an evaluation. But there is still a long way to go. For example, sensors are not perfect. There will be measurement error, and hardware malfunction, and transmission errors, and drifting. An algorithm that performs perfectly when the data received is flawless, but crashes when confronted with some imperfect data is going to be useless in the real-world setting, which is why the evaluations need to reflect actual conditions. Algorithms which would perform best in the real world should also be the ones that stand out during testing. And unless realism, such as imperfect data, is explicitly built into the evaluation, the results can never be guaranteed to accurately reflect real-world performance.

- **Use Both Real and Simulated Data** It seems that, given the types and quantity of data available today, that a combination of actual historical data and simulated data is necessary for a thorough evaluation. Each type has its benefits and can challenge the algorithms a different way.

Real sensor data is desirable, as it intrinsically reflects an actual water distribution system's water quality. As it is collected from sensors placed in the system and transmitted back to the utility, it is identical to the data that will be fed into the algorithms in a CWS. Though no data has yet been collected during a contamination event, this data can be used to calculate the false positive rates of the algorithm, as any alarm indicated while processing this

normal data would be a false positive. Based on the availability of data, data from several networks, and several sensor locations within each network, should be used to test the data. This ensures a more complete picture of the algorithms' ability to learn and recognize normal network conditions. Note that the algorithms will have to be trained (and thresholds adjusted) specifically for each network and sensor location, so the available data from each location should be divided into a training dataset and a testing set.

Actual sensor data is not enough to thoroughly test the algorithms, however, as it does not contain events to detect. Therefore, time series that contain contamination events must be simulated in order to test the algorithms' sensitivity. This can either be done by taking a real sensor time series and modifying a portion of it to simulate an attack, or by producing purely synthetic data (see section 4.1.3 for more details). No matter how they are generated, data streams should reflect a variety of attack characteristics including several contaminant types, injection locations, injection start times and durations, and injection rates. Ideally, each of the 14 contaminant classes identified in the WaterSentinel program [(USEPA), 2005e] should be represented in the testing in order to evaluate how the algorithm would perform over the whole range of recognized threats. Also, these attacks should be simulated in a variety of networks and sensor locations to show the algorithms' performance in a variety of conditions. Again, the algorithms must be trained for each network and sensor location used in the evaluation.

No matter what type of data is being used, it is important to provide enough so that accurate training can occur and the testing is valid. It is best to provide long, continuous time series of data, even several years long, as this constant data stream is what will be seen by the algorithms in a CWS. Sufficient data must be provided in order for the algorithms to learn the diurnal and seasonal variability. Therefore, at least a few years' worth of training data from each network and sensor location to be used is desirable so that the algorithms can see the patterns repeating several times and learn to identify those patterns. The testing data should also contain at least a year-long time series so that the algorithms' performance under a variety of conditions can be observed.

The ideal evaluation would include several years of both historical and simulated continuous time series. Data would be available showing hydraulic conditions and a variety of water quality parameters from multiple sensor locations in a variety of networks, as well as additional data such as 911 calls and complaint calls to the water utility. However, this quantity and quality of data is likely not available. Therefore, modifications will have to be made based on what can be obtained. It should be noted that the less data that is used to test the algorithms, the less informative and complete the results will be.

- **Current Algorithm Limitations Must Not Limit an Evaluation**

Do not let the limitations of today's technology or algorithms water down

an evaluation. As mentioned above, the key to an effective evaluation is to simulate real-world conditions. It is not to make the algorithms look good. So it is fine if imperfect data is used for the evaluation but none of the algorithms being tested have addressed this issue. It actually might be more informative to see just how the algorithms react to this new twist, and the evaluation could help the algorithm developers improve their algorithms until one is found with the flexibility, dependability, and accuracy needed for a CWS. It would be a disservice to the CWS development process to withhold available data from an evaluation because it is believed that the algorithms are not able to handle it.

- **The Evaluator’s Responsibility is to Plan, Execute, and Report on the Evaluation**

The job of the evaluation developer is to specify the details of the experiment, provide the experimental details and requirements to the participants, run the evaluation, and report on the results. This should be done methodically, thoroughly (all details of the evaluation should be provided to the participants before the evaluation and in the final report of the results), and without personal analysis or judgement. This ensures that the evaluation is fair and unbiased.

But that is all that is required of the evaluation developer. They are not responsible for algorithm performance. They are not required to adapt the testbed to accommodate individual algorithms. They are not required to

interpret the results. They are not responsible for addressing issues outside of the evaluation. No one evaluation will ever be able to address all of the issues. But in order to maintain the integrity of the study, it is important to clearly define and adhere to the experimental details, without allowing outside parties to influence the results.

- **Make Use of Available Technology**

It is beneficial to make full use of software and hardware that could be used for the evaluation, especially established technologies. This saves time in development and debugging, and the reputation of the components used can add validity to the testbed.

Chapter 7

Future Work

No evaluation is going to be the single, ultimate, final word on CWS algorithms. The evaluation and algorithm improvement process is a recursive one. Each evaluation will likely expose areas where both algorithms and the experimental design can be improved. Algorithm developers will work on their areas of weakness and will submit the new algorithm to another evaluation. Evaluators will modify their evaluation approach and plan new studies that are more realistic or relevant to an area of need. And the process will continue until all parties are satisfied that the algorithm(s) have achieved the performance necessary to be implemented in a CWS.

The real value of this thesis work is that it creates a building block for future studies to build from. While the results obtained when the evaluation is actually implemented will be interesting and will provide insights into the algorithms tested, there will be other evaluations and the algorithms will be modified. However, the lessons that have been learned and ideas that have been generated

during the design and implementation of the experimental design can be used for future, improved evaluation studies.

7.1 Continuing Evaluation Projects

This thesis work has been done in collaboration with researchers at the USEPA who are also working on the challenge of developing an effective CWS. The WaterSentinel project, described below, will build on the findings of this research in their evaluation of CWS's.

7.1.1 WaterSentinel

In 2006, the USEPA initiated the WaterSentinel Initiative in response to the Homeland Security Presidential Directive that charges the USEPA with the responsibility to "develop robust, comprehensive and fully coordinated surveillance and monitoring systems to provide early detection and awareness of water contamination" [wat, 2005] so that action can be taken to minimize the public health and economic impacts of such a contamination. The goal of the program is to design, install, and evaluate a model CWS in order to "prove the [CWS] concept so that drinking water utilities of all sizes and characteristics can adopt and implement an effective contaminant warning system" [wat, 2005].

The WaterSentinel program encompasses a wide variety of responsibilities from system architecture tasks such as setting up systems and databases for monitoring and surveillance, integrating data from water utilities with that available in the

public health surveillance sector, and adopting procedures for anomaly detection, to consequence management issues including establishing response strategies for dealing with a credible detection alert and decontaminating the water system after a contaminant has been found in it. In order to maximize the chances of timely detection of a contamination, a variety of data sources will be integrated. These include the data from the water quality sensors in the water distribution system, results of regular laboratory analysis of drinking water looking for high threat contaminants, public health data such as prescription sales and emergency room visits, reports of customer complaints about the quality of the water (for example, if there are complaints that the water smells odd), and 911 calls [wat, 2005].

WaterSentinel will utilize the same approach described in this evaluation: a network of online water quality sensors will be placed in the water distribution system, and the data received from them will be analyzed by event detection algorithms. Recent research has suggested the potential for a number of contaminants of concern to produce detectable changes in specific water quality parameters. While there are numerous parameters that respond to contaminants, the research has found that the most effective parameters are free chlorine, total organic carbon, electrical conductivity, and pH. Oxidation/reduction potential serves to corroborate chlorine sensor results [(USEPA), 2005d].

Free chlorine, total organic carbon, electrical conductivity, pH, and oxidation reduction potential sensors will be used in the pilot utilities for the WaterSentinel study. Event detection algorithms will work in near real-time by receiving these sensor signals via remote telemetry through the SCADA system, analyzing them

in near real-time, and returning their result. Various event detection algorithms are currently being developed by researchers and commercial vendors, though many of the approaches are still early in the research and development phase and only a few are available commercially at this time. The WaterSentinel Initiative may deploy one or more event detection systems at the pilot utilities. However, given the early state of the technology, it is not clear which technology should be selected for deployment or how to evaluate the various technologies available commercially or in the development process.

WaterSentinel has proposed to perform an evaluation of CWS in 2006 to select a small subset of CWS's to deploy at the first pilot utility. The evaluation plan is still under development, and there are many open questions about the desired performance characteristics of a CWS. For instance, what is the acceptable level of false positives? What other performance measures are important and how exactly should they be calculated? Should a CWS detect only contamination events, or should it be used to identify water quality anomalies unrelated to contamination? How soon does a CWS need to detect an event in order for that information to be useful? How much and what type of data is needed to characterize the performance of CWS?

7.1.2 TTEP

The USEPA has developed the Technology Testing and Evaluation Program (TTEP) to help achieve the goals outlined in the WaterSentinel project, specifically the

analysis and comparison of various event detection strategies. The mission of TTEP is to "provide reliable information regarding the performance of homeland security related technologies" [(USEPA), 2005b] and to prepare objective, third-party information on CWS technologies that can be used by those seeking to purchase and use the technologies. TTEP does a variety of rigorous tests to determine the performance of primarily commercially available products. Event detection technologies is one of the main areas of focus for this project.

7.2 Testbed Improvements

As stated several times throughout this thesis, the goal of an effective algorithm evaluation is to simulate, as accurately as possible, conditions which the algorithm will encounter if implemented in a CWS. There are many improvements that could be made to the evaluation plan described here (section 4.1.4) in order to make the conditions more realistic. For example, the algorithms were given unlimited time in this study to analyze the data, make their status decision, and request the next piece of data. This will not be the case in a real system: the sensors will continue to send water quality readings in real-time, and the algorithms must be able to handle this constant stream of data. Also, the data used in this project was not in the SCADA format, the format that is used by most water distribution systems.

7.2.1 ASSET

In order to address these testbed issues, the WaterSentinel program has initiated the development of the Advanced Simulation and SCADA Evaluation Tool (ASSET) for algorithm evaluation. ASSET will be a testing tool that aids with all aspects of algorithm testing. It will contain a database of historical and simulated datasets (though the user will have the option to use their own data or even hook the system up to actual sensors), several event detection algorithms (though again the user can add their own algorithms), and various visualization and analysis tools. These will all be integrated into a user-friendly analysis tool, which can easily be configured to test the desired algorithm(s) using the desired data, and then provide the requested reports on the results.

The main goal of the ASSET tool is to allow algorithms to be tested in an environment like that they would see in the field. The plan is to develop a seamless architecture, so that the algorithms will not know if the input they are receiving is coming from physical sensors or from a database of stored data. The data will be fed in real time, and the testbed will be based on industry SCADA and database standards. In fact, it has been suggested that this ASSET tool could also be part of the final CWS: simply choose the sensor data as input, select which algorithm(s) to use to analyze the data, and ASSET will provide real-time analysis and visualization of the data. When completed, this tool will greatly improve evaluation quality.

7.3 Data Improvements

Even a great evaluation tool like the one described above is useless without quality data feeding into it. Evaluations obviously depend critically on the availability of or generation of statistically realistic event and non-event data streams. The ability to generate a large volume of simulated data will increase greatly with the completion of the MCSim / MS-Epanet tool described in section 4.1.3. However, even with that tool, there is still much to consider when assembling a dataset for algorithm evaluation.

7.3.1 Analysis Using Real Data

Obviously, actual historical data is desirable for testing algorithms, as it inherently gives an accurate example of water quality conditions. As it comes from actual sensors in the distribution system, it automatically contains the imperfections that have yet to be addressed in simulation such as sensor error and missing data.

More of this data is necessary, however, if it is to be used for evaluation. There are already programs in place to collect a large volume of continuous data from a variety of water quality sensors in pilot utilities. This data will be very useful in testing the algorithms, and especially in establishing the false positive rates (as the data collected so far contains no events).

7.3.2 More Realistic Simulations

Simulated data is an excellent alternative to historical data, and is certainly more flexible. Realistic attacks can be simulated, and large volumes of data can be generated using a tool like the one described in section 4.1.3. Below are some factors to consider in order to add realism to generated data and to create a comprehensive set of benchmark data.

- Network Variables: Obviously a benchmark dataset would include data from a variety of networks and a variety of conditions within those networks.
 - Network Size and Layout: It is important to include model networks that vary in size, as CWS's are being designed to work in both very small and very large water distribution systems. A variety of network layouts should also be included, from those that are basically straight with rarely changing flow patterns, to networks that contain main loops, pumps, and sources in which hydraulics are constantly changing.
 - Network Hydraulics: As mentioned above, it will be desirable to include networks that have a variety of hydraulic conditions. But the hydraulics can also be altered within a network by changing or randomizing pumping patterns, opening and closing valves, and modifying rules for tanks and reservoirs.
 - Network Demands: The constantly variability and daily and seasonal fluctuations in water demands through any network is a major area

of interest. Algorithms must see these fluctuations in order to learn to recognize normal variability. Stochastic demand models are being developed, and both total network and individual nodal demands will need to be randomized in order to reflect the wide variety of conditions that are possible in a network. It would also be beneficial to simulate extreme demand conditions that would affect the system but would not be considered an event, such as those caused by a fire.

- **Sensor Variables:** Data will also be affected by which sensors are used, where they are placed, and how they are run.
 - **Number of Sensors:** The number of sensors placed in a network will obviously affect the performance of event detection algorithms. The more sensors that are deployed in a network, the more likely that a given attack will cause a detectable water quality change in at least one of them. However, budgeting requirements limit the number that can be used, and care must be used to choose a realistic number of sensors to evaluate the algorithms with so that the detection results are credible.
 - **Location of Sensors:** The problem of placing the available sensors in the most useful locations is described in section 3.5. Changing where the sensors are located will also affect how well algorithms can detect a network attack.
 - **Types of Sensors:** There are a countless number of water quality sensors

available, designed to monitor a wide range of water quality parameters. The ones that are chosen will determine which types of attacks can be detected, as different contaminants react differently with different water quality parameters. It has been found that free chlorine, TOC, conductivity, and pH are the most effective in detecting a wide range of contamination events [(USEPA), 2005d], however other sensors such as temperature and turbidity sensors are also used by water utilities.

- Sensor Sensitivity and Accuracy: The sensitivity and accuracy of a sensor will greatly affect the quality of the data produced. Building in realistic errors in the sensor measurements will help accurately replicate actual network conditions.
- Sample Frequency: The frequency with which data is collected by the sensors and transmitted to the processing and analysis center will certainly affect the CWS's performance. If sensors are polled too often, it will create such a large volume of data that the data processor will likely be unable to process it all effectively. However, if the polling is too infrequent, it is likely that an event could go past a sensor between readings and never be detected. A balance between these two factors will need to be achieved. It is important to accurately reflect what will be seen in the system, so if the water distribution collects data every five minutes, the evaluation dataset should reflect this.
- Attack Variables: Finally, the specifics of the simulated attacks will chal-

lengthen the algorithms.

- Attack Duration and Timing: The length and start time of an attack will greatly affect the amount of damage that is caused. For example, a one-hour attack at 7 a.m. will move through the system much faster than a five-hour attack beginning at noon. Again, simulating attacks with a variety of durations and start times will be helpful in training and thoroughly testing the algorithms.
- Contaminant: Another factor that will influence an attack's appearance to the sensors is the contaminant chosen. Some contaminants react strongly with chlorine and quickly and drastically change the water quality, while others have little to no effect on most water quality parameters. Choosing a variety of contaminants with a wide range of chemical reaction characteristics is important in truly determining the algorithms' abilities.
- Attack Location: Simulating attacks from a variety of network locations will also create an assortment of attack signatures. The location of the attack will affect which sensors the attack reaches, how long it takes to get there, and how long it takes to pass by. All of these factors affect detection, and should be considered.
- Attack Type: All of the items above assume the same type of attack that has been considered in all of the literature read: the continuous injection of a contaminant at a single network node. While this is a

likely attack scenario and certainly should be examined closely, it would also be beneficial to consider other possibilities as well. For example, simulate what would happen if attackers used bombs to burst open main network pipes. Also include surprise attacks in the evaluation that are unlike any in the training set in terms of contaminant used or timing.

Note that whichever data source is used for an evaluation, the more data used, the better. So provide any available information to the algorithms including sensor data measuring multiple parameters, tank water level and pump status data, information on water flow and nodal demands, and any outside data sources such as 911 calls. Even though not all algorithms will be able to incorporate all of the data into their calculations, the information is there for those that can, and the probability of accurate and quick detections is maximized.

7.4 Improved Data Analysis

Once the evaluation is executed and the data is collected, the evaluator must decide on how to quantify and report on the results. Several ideas (most fairly standard for event detection algorithm evaluations) are described in section 4.7.2, and more ideas are presented here.

7.5 Additional Performance Measures

The following is a list of possible performance measures to use when evaluating and comparing event detection algorithms to determine which would be best for a given water distribution system. The specific measures chosen, and the weight given to each one, will depend on the needs and capabilities of each individual utility.

- Health impacts before detection: Certainly, one of the main reasons for developing CWS's is to minimize the public health effects from possible attacks. Therefore, one measure would be to compare the average number of people affected by a contaminant before an algorithm detects the attack. The Threat Ensemble Vulnerability Assessment tool (TEVA) developed by the USEPA [Murray et al., 2004] is a possible tool that could be used to quantify this, as it will calculate the number of people who are infected, diseased, and killed by a described attack scenario in a water distribution system. This would be a more powerful way, perhaps, of comparing the effectiveness of algorithms.
- Cost-benefit Analysis: Petty et al. [Petty et al., 2000] discuss a new method for comparing automatic incident detection algorithms (AID's), specifically in the traffic incident detection domain. They note the drawbacks of the current approach of measuring detection rates versus false alarm rates, especially the fact that normal ROC curves treat all events equally, with the detection of a small local attack receiving the same importance as the early

detection of a large-scale assault.

Their solution is to compare the benefits of a given algorithm, which in this case would be the lives and money saved if an attack is quickly detected, to the cost of implementing an AID. To do this, the cost of implementing the AID, including hardware and installation costs, maintenance costs, and the costs associated with verifying alerts (the cost of sending out a crew, collecting samples, have the samples analyzed at a lab, ...) is compared to the cost benefit received by using the system. In the case of a water distribution system, this would include the money saved by not having to clean the system and prepare it for continued use, medical costs for treating infected citizens, and the cost of lives lost. This approach "allows flexible formulations of cost" [Petty et al., 2000], and the cost / benefit function can be modified to meet a system's particular needs and characteristics. Once these costs are defined, however, they can be used to train algorithms to the desired false positive / false negative rates as well as to provide a direct comparison between AID's.

This type of analysis will certainly be important when actually choosing a CWS to implement. The potential systems will certainly have a wide variety of installation and running costs associated with them. Also, the cost benefit of having a particular system could easily be quantified by using a program such as the TEVA tool described above. This could more accurately quantify the actual societal benefit that each potential CWS would provide.

- F-measure: The F-measure has been mentioned as an attractive way of comparing algorithms because it rates each algorithm using a single value [(USEPA), 2005a]. This value is the harmonic mean of the sensitivity and specificity and is given by:

$$f = \frac{2 \cdot \text{sensitivity} \cdot \text{specificity}}{\text{sensitivity} + \text{specificity}} \quad (7.1)$$

- Acceptability: The acceptability of a CWS considers how well it meets the need of the utility. For example, is the system and its data, monetary, and personnel requirements agreeable to all stakeholders? Do they believe the system is valuable, and are they willing to cooperate fully in its execution? The CDC group define acceptability as "the willingness of participants and stakeholders to contribute to the data collection and analysis", and they note that "acceptability can vary over time as the threat level, perceived value of early detection, support for the methods of surveillance, and resources fluctuate"[James W. Buehler et al., 2004].
- Effectiveness: Obviously, the effectiveness of a CWS is of utmost importance. In addition to the performance measures described in section 4.7.2, the following can be used to discuss how effective a system is.
 - Contaminant Coverage: How many of the contaminants of interest can the system detect? The water quality parameters used and the ability to detect attacks that influence water quality parameters quickly versus

slowly will determine this.

- Resilience to Imperfect Data: Does the algorithm still run well if there is faulty or missing data? Or does it crash?
- Resilience to Stress: Can the system operate even at high computing levels?
- Flexibility: The CDC working group notes that a CWS's "adaptation to changing detection needs or operating conditions should occur with minimal additional time, personnel, or other resources" [James W. Buehler et al., 2004]. Flexibility includes the ability to add a data stream if more information becomes available, to change the detection algorithm if a better one is developed, and to adjust the algorithm or thresholds to account for changing data. Measures associated with flexibility include the following.
 - Adaptability: How does the algorithm handle new data or changes in the data (for example, if gradually increasing water demands affect the water quality baseline)? Does it need to be re-trained, or can it adapt by itself?
 - Scalability: Will the system be able to adapt if additional sensors or input data streams are changed? What changes or re-training would need to be done to accommodate a new sensor?
 - Transparency: Is it clear how the algorithm works? Can parameters be adjusted if need be? Or is the system just a black box that takes input and gives and output?

- Upkeep: Does the system need regular maintenance and recalibration?
If so, does the algorithm initiate this, can the water utility do it, or is a company representative needed?
- Portability: The portability of a system describes "how well the system could be duplicated in another setting" [James W. Buehler et al., 2004]. Factors that affect the portability of a system include:
 - Compatibility: Is the CWS compatible with what is already being used in the utility? Will the event detection algorithm work with any data stream, or is it necessary that it be paired with the company's hardware? Will it be necessary to buy specialized hardware and software in order to operate the system?
 - Customization: Can the algorithm parameters and output display be customized for a particular utility?
 - Ease of Use: How easy is the system to use? Will operators require extensive training to be able to read and interpret the algorithm output, or is it fairly self-explanatory?
 - Installation: Can the system be installed by the water utility, or must a product representative set it up? What hardware must be bought in order to use the system?
- Stability: Stability measures a system's ability to be run continuously without disruption? The CDC group suggests measuring this by "the frequency

of system outages or downtime for servicing during periods of need, including downtime of data providers, the frequency of personnel deficiencies from staff turnover, and budget constraints” [James W. Buehler et al., 2004].

- **Product Assurance:** Certainly, a water utility wants assurance that all portions of an installed CWS are quality products that have been thoroughly tested. For example, does the product come from a reputable company? Has the product been thoroughly tested (including in the field) and shown to work? Is there a warranty or product support provided by the producer?
- **Software Issues:** What are the memory, CPU, and disc requirements for the software?

7.6 Improved Figures

ROC curves and DET curves were proposed for use in this evaluation (described in sections 4.7.3 and 4.8). However, the following offer some more options to consider when creating figures for an evaluation report.

7.6.1 Volume Under a Time-ROC Surface

The construction of ROC curves is a standard evaluation approach for analyzing outbreak detection methods. However, the information they provide about sensitivity and specificity does not necessarily give an accurate picture of the algorithms’ performance. Kleinman and Abrams point out that ”sensitivity and speci-

ficity need to be supplemented by the timeliness: a method with perfect sensitivity and specificity that detects outbreaks too late is useless” [Kleinman and Abrams, 2005].

They propose to incorporate time into the ROC curve by making it three-dimensional, with the third dimension representing detection time. Similarly to how area under the ROC curve is used to compare algorithms, volume under the time-ROC surface (VUTROCS) is used to determine an algorithm’s effectiveness. A larger VUTROCS indicates better specificity, sensitivity, and timeliness. Methods for constructing a time-ROC surface are described in their work [Kleinman and Abrams, 2005].

VUTROCS could certainly be applicable to this field. It would be straightforward to generate the time-ROC surfaces using the instructions provided by Kleinman and Abrams, and the resulting figure would provide much more information than the normal ROC curve and would be a great way to obtain a single evaluation value that combines detection rates, false alarms, and detection time.

7.6.2 AMOC curves

AMOC (Activity Monitor Operating Characteristic) curves [Fawcett and Provost, 1999] have been developed specifically for the evaluation of activity monitoring algorithms, or algorithms that process large quantities of data to find unusual events. The x-axis of an AMOC curve shows a normalized false alarm rate, such as the number of false alarms per day. The expected value of the a user-defined score function is plotted on the y-axis. Equation 7.2 gives a possible score function that

only gives credit for detections made within two hours.

$$s(\tau, \alpha, D) = \begin{cases} 0 & \text{if } 0 \leq \alpha - \tau \leq 2 \\ 1 & \text{if } \alpha - \tau > 2 \end{cases} \quad (7.2)$$

where α gives the time when an alarm is raised by the algorithm, τ is the time associated with the beginning of the attack, and D represents a particular time series. Note that $\alpha - \tau$ is the detection time.

Obviously, the shape of the AMOC curve varies depending on the definition of the score function. This flexibility to define a scoring function based on the needs of an individual utility is not available with a standard ROC curve.

7.6.3 Histograms

A popular class of figures is histograms, which plot a desired measure versus the frequency which with that occurs. Several histograms could be used illustrate algorithm(s) performance, including plots with the number of timesteps to detection, the number of attacks detected within n timesteps, or the median number of false positives per scenario as the measure plotted on the x-axis.

7.6.4 Additional Plots

There are countless plots that can be made using the algorithms' output, and it must be decided which ones provide the most information regarding the purpose of the evaluation. Some additional ideas for plots not yet mentioned include a

plot of detection time versus the percent of the population affected (useful if the benefit of the algorithm is being considered) and a plot of the detection time versus the false alarm rate.

7.7 Algorithmic Improvements

As evaluators work to improve their testing methods, algorithm developers will also be modifying their algorithms to make them more accurate, more flexible, more efficient, and to reduce false positives. Cooperation between evaluators and algorithm developers will certainly help to expedite the process of developing an algorithm that fulfills the needs of a CWS. For example, a post-evaluation dialogue, such as the one conducted as part of the DARPA evaluation (see section 3.2.2) could help to uncover the weaknesses and potential of each algorithm.

Several suggestions have been made for how to improve the algorithms. Most of these suggestions involve pushing the algorithms forward and training them on more complex data. For example, full years of data could be fed to the algorithms so that they could more completely learn seasonal changes. Or imperfect or incomplete data could be used to observe how the algorithm will react and make any necessary adjustments. While these types of changes are certainly valuable, they do not require drastic modifications to the overall algorithmic approach. However, there are some suggested improvements that would require more extensive changes to the algorithms.

- Use of Multiple Data Streams: One of these desired advancements is to

integrate multiple data streams into the algorithm analysis. For example, an algorithm may receive real-time data from multiple sensor locations, each with several sensors measuring a variety of water quality parameters. It could also be receiving data streams describing 911 calls, doctor visits, and calls to the water utility complaining of water quality.

It would certainly be valuable, would likely provide more confidence in the results, and would reduce the number of false positives if an algorithm could consider all of these factors when making a decision about the status of the system. This could be done in two ways. One possibility would be to have the data from all of the data streams as input parameters into the algorithm, and this vector could be used to classify the system condition as normal or abnormal. The other option would be to analyze each data stream separately and then combine the results. For example, the following statement could be used to decide whether or not to sound a network-wide alarm in a hypothetical system with four data streams (data from one pH sensor, two chlorine sensor streams, and data from 911 calls,):

$$\begin{aligned}
 P(TP) &= P(event|pHSensor1 = alarm) \cdot P(event|chSensor1 = alarm) \cdot \\
 &\quad P(event|chSensor2 = normal) \cdot P(event|911Calls = normal)
 \end{aligned}
 \tag{7.3}$$

where $P(TP)$ is the probability of the alarm being a true positive. Therefore,

this statement says that the probability that an attack has actually occurred in the network is the product of the probabilities that there is an attack given the current value from each individual data stream (the analysis of the pH and first chlorine data streams indicating abnormal conditions, but nothing unusual about the second chlorine sensor data or 911 calls). The individual probabilities would have to be calculated by evaluating the effectiveness of the analysis of each data stream individually.

”When these information streams are integrated and used to evaluate a possible contamination event, the credibility of the incident can be established more quickly and reliably than if any of the information streams were used independently” [(USEPA), 2005e]. Clearly, there is nothing to lose and everything to gain by using as many data streams as are available. Hopefully, algorithms will be able to integrate the different data to provide more reliable detection.

- **Integration of Spatial and Hydraulic Information:** In addition to the analysis of multiple data streams, it has also been suggested that spatial, hydraulic, and systems operation information be used by CWS algorithms in addition to the water quality data. This data can be used to establish the relationship between the sensors (for example, knowing that water is flowing from sensor a to sensor b in about two hours) and to identify when water quality changes can be attributed to explainable system operations (for example, when a pump turns on or off). This would certainly reduce the number of alarms,

as a large number of potential alerts could be eliminated if their hydraulic origin could be identified by the algorithm. Systems operators would then likely pay more attention to the alerts that did occur, knowing that the water quality changes could not be attributed to normal systems operations.

- **Determination of Attack Characteristics:** The experimental plan proposed in this work only requires the algorithms to determine if current conditions are normal or abnormal. However, the algorithms could certainly attempt to give more information about the attack. The DARPA evaluation of computer network security algorithms (described in section 3.2.2) required participants to provide the date and time of the supposed attack, as well as the address of the computer where the attack was believed to have occurred and a confidence score (between 0 and 1) indicating how certain the algorithm was that the alarm was correct [Lippmann et al., 2000a]. Algorithms also had the option to identify what type of attack had occurred, chosen from a set list of attack categories.

The time of the attack is obviously the fundamental output needed from the algorithms, and it is provided indirectly in this evaluation framework as the first timestep for which the algorithm signals abnormal conditions. In addition, the challenge of estimating the location of the attack has received much attention lately. Source inversion is the problem of using time-sensitive concentration levels from sensor locations to identify the time and location that an attack occurred. Note that this is the process of using

the data from *events* at the individual sensors to determine where and when the original *attack* occurred. An initial, simple algorithm for doing this is to simulate an attack at each possible source and to see which produces the water quality signature closest to the one observed in the system [Laird et al., 2005b]. However, more complex and efficient methods are being developed [Laird et al., 2005a]. This ability to determine the likely source of observed abnormal conditions would certainly be useful, as knowing where to go would make attack verification and containment efforts much easier.

The other possible information that an algorithm could attempt to identify is the type of contaminant injected. This would require testing of a wide range of contaminants to see how they affect the various water quality parameters. Then the algorithms could try to fit the current water quality conditions into one of the predetermined attack classifications. This capability would be interesting and could provide useful information for how to react to the alarm (for example, if a HAZMAT (hazardous materials) crew must be brought in).

- Use of Algorithms to Identify Water Types: One final, more theoretical wish, is to use this algorithm development, especially with algorithms that train themselves using machine learning techniques to cluster or classify water conditions, to attempt to define various 'normal' water states. "An important factor in identifying the background water quality within a distri-

bution system is the determination of whether or not the data are influenced by an underlying trend. The recognition of temporal trends in key water quality variables is one of the ultimate goals of any water quality monitoring program” [Srinivasan et al., 2005]. One draw of AI techniques is their ability to extract patterns that are not obvious from data. Perhaps these algorithms will help researchers identify classes of ‘normal’ water that were previously not understood.

7.8 Other Implementation Details

Section 3.6, which discusses sensor networks, identifies several difficulties will eventually have to be addressed before a complete CWS can be installed in a water distribution system. These include the following:

- **Sensor Issues:** These include the logistics of positioning and accessing sensors in the network, as sensors will need to be recalibrated, inspected regularly to ensure proper functioning, and replaced occasionally. Also, the sensors need to have some power source, and drift needs to be considered since they will be in the presence of flowing water.
- **Data Transmission:** Getting the data back to the data processing center is an easily overlooked difficulty. Radio transmission is often used, but it is important that the sensors have the hardware capabilities of sending a signal that will reach the desired target. Faulty connections and interrupted transmissions are also considerations, as they will cause missing data.

- Data Storage: If there are multiple sensor locations, each with more than one water quality sensor, all regularly broadcasting data back to a central location, data management and storage is an issue, as a large volume of data will be arriving at a high speed. A flexible, scalable, easily accessed data storage method must be developed in order to deal with the data when it comes in, and be able to access it later.

Chapter 8

Summary

The need to protect the Nation's municipal water systems is great, as the water supplies' vulnerability and ability to reach all citizens makes them prime terrorist targets. In the last few years, as the awareness of terrorist threats has been heightened, attention has been focused on developing Contaminant Warning Systems (CWS's) to be placed into water distribution systems. The proposed CWS design consists of a network of continuous, real-time water quality sensors, placed through the distribution network and continuously collecting water quality data. The data received is then transmitted to a processing center and input into a event detection algorithm, which analyzes the data to determine whether conditions are normal or if an attack is suspected. If the algorithm suspects a problem, further action will be taken to determine if an attack truly has occurred. This may include sending a team out to sample and test the water in the area of concern and tapping other data sources such as 911 calls and emergency room visits. If it is found that the alarm is credible, containment and cleanup efforts will begin that

may include flushing the system and alerting the general public to stop using the water.

There are several aspects of the system that need to be developed and tested before a CWS is ready to be deployed and used in a water distribution system. Reliable and accurate water quality sensors need to be developed, and the optimal placement of these sensors in a system needs to be determined. Methods for transmitting, collecting, and managing the data need to be established. Reliable and accurate detection algorithms need to be developed to analyze the data and decide whether or not further action needs to be taken. And guidelines for responding to an algorithm alert need to be made.

This work seeks to initiate assessment efforts in the event detection algorithm domain. A literature review was completed in order to learn from similar studies of event detection algorithms in other fields, such as public health surveillance. An evaluation plan was developed to measure algorithms' abilities to detect abnormal conditions in a stream of water quality data. This experimental framework was then presented to a variety of researchers in this field and their feedback was obtained. Finally, the lessons learned were used to create a general framework and provide suggestions for future evaluations of this kind.

Hopefully the work done and lessons learned here will help move research in this area forward, and eventually an algorithm will be developed with the accuracy and reliability needed to be implemented in a CWS.

Bibliography

- [Bio, 2004] (2004). Evaluation of algorithms for outbreak detection using clinical data from five u.s. cities. Technical report, Potomac Institute for Policy Studies and General Dynamics Advanced Research Center and IBM Watson Research Center and Johns Hopkins University Applied Physics Laboratory and Stanford University and University of Pittsburgh and Walter Reed Army Institute for Research. http://www.syndromic.org/publications/5_cities_eval_final.pdf.
- [wat, 2005] (2005). Proposed water sentinel initiative. <http://www.epa.gov/nhsrcc/pubs/fsWaterSentinel062005.pdf>.
- [nor, 2006] (2006). Normnet. <http://www.fti-net.com/NormNet.asp>.
- [Abadi et al., 2005] Abadi, D. J., Madden, S., and Lindner, W. (2005). Reed: Robust, efficient filtering and event detection in sensor networks. In *Proceedings of the 31st VLDB Conference*, Trondheim, Norway.
- [Aggarwal, 2005] Aggarwal, C. C. (2005). On abnormality detection in spuriously populated data streams. In *Proceedings of the ACM SIAM Conference on Data Mining*. ACM),.

- [Akyildiz et al., 2002a] Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002a). Wireless sensor networks: a survey. *Computer Networks*, 38(1):393–422.
- [Akyildiz et al., 2002b] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002b). A survey on sensor networks. *IEEE Communication Magazine*, 40(8):102–114.
- [Berry et al., 2004] Berry, J., Hart, W., Phillips, C., and Uber, J. (2004). A general integer-programming-based framework for sensor placement in municipal water networks. In *Proceedings of the 2004 World Water & Environmental Resources Conference*, Salt Lake City, UT. ASCE.
- [Buckeridge et al., 2005] Buckeridge, D. L., Burkom, H., Campbell, M., Hogan, W. R., and Moore, A. W. (2005). Algorithms for rapid outbreak detection: a research synthesis. *Journal of Biomedical Informatics*, 38:99–113.
- [Buckeridge et al., 2004] Buckeridge, D. L., Burkom, J., Moore, A., Pavlin, J., Cutchis, P., and Hogan, W. (2004). Evaluation of syndromic surveillance systems - design of an epidemic simulation model. *Morbidity and Mortality Weekly Report (MMWR)*, 53:137–143.
- [Bulut et al., 2005] Bulut, A., Singh, A. K., Shin, P., Fountain, T., Jasso, H., Yan, L., and Elgamal, A. (2005). Real-time nondestructive structural health monitoring using support vector machines and

wavelets. <http://ieng9.ucsd.edu/~lyan/publications/2005NDE.pdf#search=ahmet%20bulut%20realtime%20nondestructive>.

[Burbeck and Nadjm-Tehrani, 2004] Burbeck, K. and Nadjm-Tehrani, S. (2004). Advice- anomaly detection with real-time incremental clustering. <http://www.ida.liu.se/~rtslab/publications/2004/burbeck04advice.pdf#search=ADWICE%20Simin>.

[Burkom et al., 2005] Burkom, H., Elbert, Y., and Thompson, M. (2005). Using point-source epidemic curves to evaluate alerting algorithms for biosurveillance. In *Proceedings of the 10th Biennial CDC and ATSDR Symposium on Statistical Methods*.

[Burkom et al., 2004] Burkom, H. S., Hutwagner, L. C., and Rodriguez, R. T. (2004). Using point-source epidemic curves to evaluate alerting algorithms for biosurveillance. In *Proceedings of the 2004 American Statistical Association, Statistics in Government Section*, Toronto, Canada. American Statistical Association.

[Chan et al., 1999] Chan, P., Fan, W., Prodromidis, A., and Stolfo, S. (1999). Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems*, pages 67–74.

[Dodd et al., 2006] Dodd, M. C., Vu, N. D., Armann, A., Le, V. C., Kissner, R., Pham, H. V., Cao, T. H., Berg, M., and Gunten, U. V. (2006). Kinetics and mechanistic aspects of as(iii) oxidation by aqueous chlorine, chloramines,

- and ozone: Relevance to drinking water treatment. *Environmental Science Technology*, 40:3285–3292.
- [Dutta et al., 2005] Dutta, P., Grimmer, M., Arora, A., Bibyk, S., and Culler, D. (2005). Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks*, Los Angeles, CA. IPSN.
- [Estrin et al., 1999] Estrin, D., Govindan, R., Heidemann, J., and Kumar, S. (1999). Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the 5th Annual International Conference on Mobile Computing and Networks*, Seattle, WA. MobiCOM.
- [Fawcett and Provost, 1999] Fawcett, T. and Provost, F. (1999). Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM.
- [Flexer,] Flexer, A. Statistical evaluation of neural network experiments: Minimum requirements and current practice. Technical Report oefai-tr-95-16, The Austrian Research Institute for Artificial Intelligence.
- [Haines et al., 2001a] Haines, J., Lippmann, R., Fried, D., Zissman, M., Tran, E., and Boswell, S. (2001a). 1999 darpa intrusion detection evaluation: Design and procedures. Technical report. <http://www.ll.mit.edu/IST/ideval/pubs/2001/TR-1062.pdf>.

- [Haines et al., 2001b] Haines, J., Rossey, L., Lippman, R., and Cunningham, R. (2001b). Extending the darpa off-line intrusion detection evaluations. In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition II*. IEEE.
- [Hutwagner et al., 2003] Hutwagner, L., Thompson, W., Seeman, G. M., and Treadwell, T. (2003). The bioterrorism preparedness and response early aberration reporting system (ears). *Urban Health*, 80:i89–i96.
- [James W. Buehler et al., 2004] James W. Buehler, M., Richard S. Hopkins, M., J. Marc Overhage, M., Daniel M. Sosin, M., and Van Ton, M. (2004). Framework for evaluating public health surveillance systems for early detection of outbreaks. *Morbidity and Mortality Weekly Report (MMWR)*, 53:1–11.
- [Jiao,] Jiao, B. Fedl: Formalized event description language for wireless sensor networks. Technical Report 1, CS Department, Univeristy of Virginia. http://www.cs.virginia.edu/~bj3r/FEDL_techreport.doc.
- [Johnson, 2002] Johnson, D. S. (2002). A theoretician’s guide to the experimental analysis of algorithms. In *Proceedings of the 5th and 6th DIMACS Implementation Challenges*. American Mathematical Society.
- [Jonkergouw et al., 2004] Jonkergouw, P., Khu, S., and Savic, D. (2004). Chlorine: A possible indicator of intentional chemical and biological contamination in a water distribution network? In *Proceedings of the 2nd International IWA*

- Conference on Automation in Water Quality Monitoring*, Vienna, Austria. International Water Association.
- [Kleinman and Abrams, 2005] Kleinman, K. and Abrams, A. (2005). Evaluating outbreak-detection methods using simulations: Volume under the time-roc surface. Technical report.
- [Krishnamachari and Iyengar, 2004] Krishnamachari, B. and Iyengar, S. (2004). Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor networks. *IEEE Transactions on Computers*, 53(3):241–250.
- [Kroll, 2005] Kroll, D. (2005). Water distribution monitoring: Opportunities and challenges for enhancing water quality and security. Technical report.
- [Kroll and King, 2005] Kroll, D. and King, K. (2005). Validation and testing of the operational effectiveness of an on-line security platform for the water distribution system.
- [Laird et al., 2005a] Laird, C. D., Biegler, L. T., and van Bloemen Waanders, B. G. (2005a). A mixed integer approach for obtaining unique solutions in source inversion of drinking water networks.
- [Laird et al., 2005b] Laird, C. D., Biegler, L. T., van Bloemen Waanders, B. G., and Bartlett, R. A. (2005b). Contamination source determination for water networks. *Journal of Water Resources Planning and Management*, 131:125–134.

- [Li et al., 2002] Li, D., Wong, K. D., Hu, Y. H., and Sayeed, A. M. (2002). Detection, classification, and tracking of targets in distributed sensor networks. *IEEE Signal Processing Magazine*, 19(2):17–30.
- [Li et al., 2003a] Li, S., Son, S. H., and Stankovic, J. A. (2003a). Event detection services using data service middleware in distributed sensor networks. In *Second International Workshop on Information Processing in Sensor Networks*, pages 502–517, Palo Alto, CA. IPSN.
- [Li et al., 2003b] Li, X., Kim, Y. J., Govindan, R., and Hong, W. (2003b). Multi-dimensional range queries in sensor networks. In *Proceedings of the SenSys Conference*, Los Angeles, CA. ACM.
- [Lippmann et al., 2000a] Lippmann, R., Haines, J. W., Fried, D. J., Korba, J., and Das, K. (2000a). The 1999 darpa off-line intrusion detection evaluation. *Computer Networks*, 34:579–595.
- [Lippmann et al., 2000b] Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., Weber, D., Webster, S. E., Wyschorod, D., Cunningham, R. K., and Zissman, M. A. (2000b). Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition*. IEEE.
- [Lombardo et al.,] Lombardo, J., Burkom, H., and Pavlin, J. Essence ii and the framework for evaluating syndromic surveillance systems. *MMWR*, 53

(Suppl):159–165.

[Markou and Singh,] Markou, M. and Singh, S. Novelty detection: A review - part 1: Statistical approaches. http://www.dcs.ex.ac.uk/research/pann/pdf/pann_SS_086.PDF#search='novelty%20detection%3A%20%20a%20review'.

[Markou and Singh, 2003] Markou, M. and Singh, S. (2003). Novelty detection: A review - part 2: Neural network based approaches. *Signal Processing*, 83:2499 – 2521.

[Martin et al., 1997] Martin, A., Doddington, G., Kamm, T., Ordowski, M., and Przybocki, M. (1997). The DET curve in assessment of detection task performance. In *Proceedings of Eurospeech '97*, pages 1895–1898, Rhodes, Greece.

[McKenna et al., 2005] McKenna, S. A., Wilson, M., Cruz, V., Madueuke, N. O., and Srinivasan, S. (2005). Status report: Task 4, sandia national laboratories - epa nhsrca inter-agency agreement, fy'2005. Technical report.

[Murray et al., 2004] Murray, R., Janke, R., and Uber, J. (2004). The threat ensemble vulnerability assessment (TEVA) program for drinking water distribution system security. In *Proceedings of the 2004 World Water & Environmental Resources Conference*, Salt Lake City, UT. ASCE.

[Ostfeld and Salomons, 2003] Ostfeld, A. and Salomons, E. (2003). An early warning detection system (EWDS) for drinking water distribution systems security. In *Proceedings of the 2003 World Water & Environmental Resources Conference*, Philadelphia, PA. ASCE.

- [Ostfled and Salomons, 2004a] Ostfled, A. and Salomons, E. (2004a). Optimal layout of early warning detection stations for water distribution systems security. *Journal of Water Resources Planning and Management*, 130(5):377–385.
- [Ostfled and Salomons, 2004b] Ostfled, A. and Salomons, E. (2004b). A stochastic early warning detection system model for drinking water distribution systems security. In *Proceedings of the 2004 World Water & Environmental Resources Conference*, Salt Lake City, UT. ASCE.
- [Petty et al., 2000] Petty, K. F., Ostland, M., Kwon, J., Rice, J., and Bickel, P. J. (2000). A new methodology for evaluating incident detection algorithms. Technical report. <http://repositories.cdlib.org/its/path/papers/UCB-ITS-PWP-2000-11/>.
- [Ren et al., 2005] Ren, S., Li, Q., Wang, H., Chen, X., and Zhang, X. (2005). Analyzing object detection quality under probabilistic coverage in sensor networks. In *Proceedings of the 13th International Workshop on Quality of Service*, Passau, Germany. WQoS.
- [Resende and Werneck, 2004] Resende, M. and Werneck, R. (2004). A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10(1):59–88.
- [Schmueli, 2005] Schmueli, G. (2005). Current and potential statistical methods for anomaly detection in modern time series: The case of biosurveillance. In *Proceedings of the International Workshop on Data Mining Methods for Anomaly Detection*, Chicago, IL. KDD),.

- [Siegrist and Pavlin, 2004] Siegrist, D. and Pavlin, J. (2004). Bio-alert biosurveillance detection algorithm evaluation. *Morbidity and Mortality Weekly Report*, 53:152–158.
- [Sosin, 2003] Sosin, D. M. (2003). Draft framework for evaluating syndromic surveillance systems. *Journal of Urban Health: Bulletin of the New York Academy of Medicine*, 80:i8–i13.
- [Srinivasan et al., 2005] Srinivasan, S., Madueke, N. O., and McKenna, S. A. (2005). Literature review of approaches for determining variability in background water quality parameters and change detection.
- [Uber et al., 2004] Uber, J., Janke, R., Murray, R., and Meyer, P. (2004). Greedy heuristic methods for locating water quality sensors in distribution systems. In *Proceedings of the 2004 World Water & Environmental Resources Conference*, Salt Lake City, UT. ASCE.
- [Uber et al., 2005] Uber, J., Janke, R., Murray, R., Meyer, P., and Umberg, K. (2005). Set covering formulation for locating water quality sensors in distribution systems.
- [(USEPA), 2005a] (USEPA), U. S. E. P. A. (2005a). Overview of event detection systems for watersentinel. http://www.epa.gov/safewater/watersecurity/pubs/watersentinel_event_detection.pdf.

- [(USEPA), 2005b] (USEPA), U. S. E. P. A. (2005b). Tech brief: Technology testing and evaluation program. <http://www.epa.gov/nhsrc/pubs/tbTTEP031005.pdf>.
- [(USEPA), 2005c] (USEPA), U. S. E. P. A. (2005c). Technologies and techniques for early warning systems to monitor and evaluate drinking water quality: a state of the art review. <http://www.epa.gov/nhsrc/pubs/reportEWS120105.pdf>.
- [(USEPA), 2005d] (USEPA), U. S. E. P. A. (2005d). Watersentinel online water quality monitoring as an indicator of drinking water contamination. http://www.epa.gov/safewater/watersecurity/pubs/watersentinel_wq_monitoring.pdf.
- [(USEPA), 2005e] (USEPA), U. S. E. P. A. (2005e). Watersentinel system architecture. http://www.epa.gov/safewater/watersecurity/pubs/watersentinel_wq_monitoring.pdf.
- [Watson et al., 2005] Watson, J.-P., Hart, W. E., and Berry, J. (2005). Scalable high-performance heuristics for sensor placement in water distribution networks. In *Proceedings of the 2005 World Water & Environmental Resources Conference*. ASCE.
- [Zanardelli et al., 2005] Zanardelli, W. G., Strangas, E. G., and Aviyente, S. (2005). Failure prognosis for permanent magnet ac drives based on wavelet analysis. In *Proceedings of the International Electric Machines and Drives Conference*.

- [Zhang et al., 2003] Zhang, J., Tsui, F.-C., Wagner, M., and Hogan, W. (2003). Detection of outbreaks from time series data using wavelet transform. *JAMIA Symposium Issue*, pages 748–752.
- [Zhang and Pehkonen, 1999] Zhang, Q. and Pehkonen, S. O. (1999). Oxidation of diazinon by aqueous chlorine: Kinetics, mechanisms, and product studies. *J. Agricultural Food Chemistry*, 47:1760–1766.

Appendix A

List of Abbreviations

ADWICE Anomaly Detection with Real-Time Incremental Clustering

AI Artificial Intelligence

AMOC Activity Monitor Operating Characteristic

ASSET Advanced Simulation and SCADA Evaluation Tool

BCD Biosurveillance Using a Change-point Detector

CDC Center for Disease Control and Prevention

CWS Contamination Warning System

DARPA Defense Advanced Research Projects Agency

DET Detection Error Tradeoff

DLL Dynamically Linked Library

EARS Early Aberration Reporting System

EWDS Early Warning Detection System

FAC Free Available Chlorine

GRASP Greedy Randomized Adaptive Search Procedure

GUI Graphical User Interface

HAZMAT Hazardous Materials

HMM Hidden Markov Models

LARIAT Lincoln Adaptable Real-Time Information Assurance Testbed

MIP Mixed-integer Program Solver

MS-Epanet Multi-species EPANET

MSPC Multiple Statistical Process Control

ROC Receiver Operating Curve

RPM Randomized Pollution Matrix

SCADA Supervisory Control and Data Acquisition

SOM Self-organizing Map

TEVA Threat Ensemble Vulnerability Assessment Tool

TOC Total Organic Carbon

TTEP Technology Testing and Evaluation Program

USEPA United States Environmental Protection Agency

WDSA Water Distribution System Analysis

Appendix B

File Types Used in the Evaluation

Several file types are used by the components of the proposed experimental framework. A sample of each type is provided here.

Figure B.1: Sample *.sim file, an input into the MCSim data generation tool.
// Example MCSim/MSEpanet Monte-Carlo simulation input file (*.SIM)

```
MonteCarlo ("MCSim.out", 1000, 56761.1164);  
// Basic Monte Carlo simulation parameters specifying the desired output file,  
//number of simulations to be run, and a seed for the random number generator  
  
// Associate probability distributions with network parameters  
Distrib (N_node1_5, Uniform, 0.75, 1.25);  
Distrib (GlobalDemandMult, PiecewiseRandom, 0.4, 1.0, 1.0, 1.6);  
Distrib (GlobalBulk, Uniform, -1.0, -2.0);  
//These lines establish the randomized demands  
  
// Define simulation ensembles (can be multiple)  
Simulation { # 1:  
// Set deterministic values of simulation parameters  
// Applied to all scenarios in ensemble  
  
// MSEpanet input file  
MSEpanetFileName = 'wds1.inp';  
  
// double pump speed in link pipe1 for all simulations in #1  
P_pipe1_12 = 1.5;  
}
```

Figure B.2: Sample *.inp file, the input file for MS-Epanet. Note that this is not a real network: a few entries have been provided under each heading to show what type of network input is required.

```
C:\Documents and Settings\Katie Umberg\My Documents\Thesis\networklms_arsenite.inp 1

[TITLE]
Sodium Arsenite-Free Chlorine interaction model

[JUNCTIONS]
;ID      Elev      Demand      Pattern
2000      360.06      1.13      1      ;      4;
2005      333.88      75      1      ;      0

[RESERVOIRS]
;ID      Head      Pattern
60063      425      ;355

[TANKS]
;ID      Elevation      InitLevel      MinLevel      MaxLevel      Diameter      MinVol      VolCurve
50009      843.9      11.159      0.0      32.1      186.0      0.0
50011      1137.1      17.945      0.0      41.9      106.0      0.0

[PIPES]
;ID      Node1      Node2      Length      Diameter      Roughness      MinorLoss      Status
2010      800332      2000      903.0      20.0      138.0      0      Open
3841      800331      3591      1010.0      16.0      138.0      0      Open

[PUMPS]
;ID      Node1      Node2      Parameters
70111      701111      701112      HEAD 1 ;Zone 6 PS #1

[VALVES]
;ID      Node1      Node2      Diameter      Type      Setting      MinorLoss
80033      800331      800332      6.0      PRV      29.762      0
80035      800351      800352      6.0      PRV      37      0

[TAGS]

[DEMANDS]
;Junction      Demand      Pattern      Category
4270      0      1      ;      0
4270      0.20      2      ;      0

[STATUS]
;ID      Status/Setting

[PATTERNS]
;ID      Multipliers
;
1      1.56      1.36      1.17      1.13      1.08      1.04
1      1.20      0.64      1.08      0.53      0.29      0.90
1      1.11      1.06      1.00      1.65      0.55      0.74

[CURVES]
;ID      X-Value      Y-Value
;PUMP:      Pump curve for Pump 70111
1      0      445
1      790      365
1      1460      120

[CONTROLS]
LINK 80042 CLOSED AT TIME 0      ;Permanently closed valve

[RULES]
RULE RPUMPTIME      ;All pumps run between 6PM and 11AM only
IF SYSTEM CLOCKTIME >= 11:00 AM
AND SYSTEM CLOCKTIME <= 6:00 PM
THEN PUMP 70111 STATUS IS CLOSED
PRIORITY 2

RULE RZ6ON      ;Zone 6 pump station
```

C:\Documents and Settings\Katie Umberg\My Documents\Thesis\networklms_arsenite.inp

1

```

IF TANK 50011 LEVEL <= 15.4 ;Lower limit
THEN PUMP 70111 STATUS IS OPEN
PRIORITY 1

[ENERGY]
Global Efficiency      75
Global Price           0
Demand Charge         0

[EMITTERS]
;Junction      Coefficient

[COORDINATES]
;Node      X-Coord      Y-Coord
2000      6075995.78      2162155.72
2005      6077209.53      2160896.60

[VERTICES]
;Link      X-Coord      Y-Coord

[LABELS]
;X-Coord      Y-Coord      Label & Anchor Node

[BACKDROP]
DIMENSIONS      6072437.10      2158770.18      6087302.64      2178456.08
UNITS      None
FILE
OFFSET      0.00      0.00

[MSVARIABLES]
BULK Chlorine
BULK AsIII
;BULK AsV
CONSTANT KB      0;6.94444E-4 ;Free chlorine bulk decay rate == 1/day
CONSTANT K2      0;6.94444E-4 ;Arsenite oxidation velocity
CONSTANT MWChlorine 70.9 ;mg/mmol free chlorine as Cl2
CONSTANT MWAsIII 74.92 ;mg/mmol arsenite as As
;CONSTANT MWAsV 74.92 ;mg/mmol arsenate as As
UNIT VOLUME LITER
UNIT TIME MINUTE

[MSPIPE]
RATE Chlorine -KB*Chlorine - K2*Chlorine*AsIII/MWAsIII
RATE AsIII -K2*Chlorine*AsIII/MWChlorine
;RATE AsV K2*MWAsV*Chlorine*AsIII/MWChlorine/MWAsIII
;RATE Chlorine -6.944E-4*Chlorine - 0.00927*Chlorine*AsIII
;RATE AsIII -0.00979*Chlorine*AsIII
;RATE AsV 0.00979*Chlorine*AsIII

[MSTANK]
RATE Chlorine -KB*Chlorine - K2*Chlorine*AsIII/MWAsIII
RATE AsIII -K2*Chlorine*AsIII/MWChlorine;
;RATE AsV K2*MWAsV*Chlorine*AsIII/MWChlorine/MWAsIII
;RATE Chlorine -6.944E-4*Chlorine - 0.00927*Chlorine*AsIII
;RATE AsIII -0.00979*Chlorine*AsIII
;RATE AsV 0.00979*Chlorine*AsIII

[QUALITY]
;Node      SpecieID      InitQual
50009      Chlorine      0.3 ;Initial tank chlorine
50011      Chlorine      0.16 ;Initial tank chlorine

[SOURCES]
;Node      Type      Quality      SpecieID      Pattern
60063      CONCEN      2      Chlorine
60063      CONCEN      1      AsIII ; PAS

```

C:\Documents and Settings\Katie Umberg\My Documents\Thesis\networklms_arsenite.inp

1

[REACTIONS]

[TIMES]

Duration	48:00
Hydraulic Timestep	0:30
Quality Timestep	0:03
Pattern Timestep	0:30
Pattern Start	0:00
Report Timestep	1:00
Report Start	96:00
Start ClockTime	8:00 AM
Statistic	NONE

[REPORT]

Status	No
Summary	No
Page	0

[OPTIONS]

Units	GPM
Headloss	H-W
Specific Gravity	1.0
Trials	40
Accuracy	0.0050
Unbalanced	Stop
Pattern	1
Demand Multiplier	1.0
Emitter Exponent	0.5
Quality	Chemical mg/L
Diffusivity	100.0
Tolerance	0.001

[MSOPTIONS]

; RELTOL	1.E-8	
; ABSTOL	Chlorine	0.0
; ABSTOL	AsIII	0.0

[END]

Figure B.3: Format of the *.tso file, which contains the results of all simulations run. Note that this is an unformatted binary file and is not readable using an ordinary text editor.

<i>nnodes</i>	int; number of network nodes
<i>nlinks</i>	int; number of network pipes
<i>ntanks</i>	int; number of network tanks
<i>njunctions</i>	int; number of network junctions (=nnodes-ntanks)
<i>nsteps</i>	int; number of simulation output time steps
<i>rstep</i>	float; simulation output reporting time step in hours
<i>fltnax</i>	float; maximum value of float data type
<i>tankindex(1),...,tankindex(ntanks)</i>	int; tank node indices
<i>nodeID(1),...,nodeID(nnodes)</i>	16-byte character strings of network node IDs
<i>(xnode(1),...,xnode(nnodes)</i> <i>(ynode(1),...,ynode(nnodes))</i>	float; (x,y) coordinates of all network nodes
<i>(fromnode(1),...,fromnode(nlinks)</i> <i>(tonode(1),...,tonode(nlinks))</i>	int; connecting node index pairs for all network links, each integer node index is in the range [1,nnodes].
<i>d(1,1),...,d(nnodes,1)</i> <i>d(1,2),...,d(nnodes,2)</i> . . <i>d(1,nsteps),...,d(nnodes,nsteps)</i>	float; water demand at each node and timestep $D(i,j)$ is the demand at node i in time step j .
<i>N</i>	int; The number of contaminant source nodes for this scenario
<i>Nindex(1), Mass Rate(1), Start Time(1), Stop Time(1)</i> . . . <i>Nindex(N), Mass Rate(N), Start Time(N), Stop Time(N)</i>	<i>Nindex</i> (int): The contaminant source nodes <i>Mass Rate</i> (float): The mass rate (mg/min) for each source node; <i>Start Time</i> (long): start time (seconds) for each source node; <i>Stop Time</i> (long): stop time (seconds) for each source node
<i>M</i> <i>Nindex(1) c(1),...,c(nsteps)</i> <i>Nindex(2) c(1),...,c(nsteps)</i> . . . <i>Nindex(M) c(1),...,c(nsteps)</i>	float; sparse format threat scenario contaminant concentrations at M nodes $nodeID(Nindex(1)), \dots, nodeID(Nindex(M))$ for all time steps.

*** The data items starting from N describe the contaminant concentrations for one scenario, and are repeated in sequence for every threat scenario run.**

Figure B.4: Format of the *.tao file, which is used by the DLL.

```

C:\Documents and Settings\Katie Umberg\My ... \Dll Files\Files Used for Testing\train.tao 1
/*The .tao file which is the input into the .dll should have the following format:

**Note that (float) Ch(R, S, T) stands for the chlorine level for scenario R, sensor S,
           and timestep T
           (float) Co(R, S, T) stands for the contaminant level at sensor S for
           timestep T in scenario R

           (int) R stands for the scenario number and is between 0 and NScenarios-1
           (int) S stands for the sensor number and is between 0 and NSensors-1
           (int) T stands for time step and is between 0 and NSteps-1

Number of timesteps (int)
Number of species being traced (including chlorine and any contaminants) (int)
Number of sensors (int)
(Sensor0) Ch(0,0,0) Co(0,0,0) Ch(0,0,1) Co(0,0,1),...Ch(0,0,Nsteps-1) Co(0,0,Nsteps-1)
(Sensor1) Ch(0,1,0) Co(0,1,0) Ch(0,1,1) Co(0,1,1),...Ch(0,1,Nsteps-1) Co(0,1,Nsteps-1)
.
.
.
(Sensor(NSensors-1)) Ch(0,NSensors-1,0) Co(0,NSensors-1,0)... Co(0,NSensors-1,Nsteps-1)
(Sensor0) Ch(1,0,0) Co(1,0,0) Ch(1,0,1) Co(1,0,1),... Ch(1,0,Nsteps-1) Co(1,0,Nsteps-1)
(Sensor1) Ch(1,1,0) Co(1,1,0) Ch(1,1,1) Co(1,1,1),... Ch(1,1,Nsteps-1) Co(1,1,Nsteps-1)
.
.
.
(Sensor(NSensors-1)) Ch(0,NSensors-1,0) Co(0,NSensors-1,0)...Co(0,NSensors-1,Nsteps-1)
.
.
.
(Sensor(NSensors-1)) Ch(NScenarios-1,NSensors-1,0),...Co(NScenarios-1,NSensors-1,Nsteps-1)

A sample input file is included below that has 3 sensors and collecting chlorine and
contaminant data for 5 time steps.

Note that for the first scenario, sensor 0 sees the chlorine level go from .8 to .5 to 7
to .6 to .3, while the contaminant is at 0.5 the whole time (indicating an event during
this time). For the same scenario, sensor 1 sees chlorine levels of .3, 0, .1, .1, and
.5, and again an event is occurring with a contaminant level of 0.5 all along. For the
second scenario, the sensor 0 registers a both a chlorine and contaminant level of .3
during the first time step...

*/

5 //number of timesteps
2 //number of species (in this case, chlorine and a contaminant)
3 //number of sensors
0 .8 0.5 .5 0.5 .7 0.5 .6 0.5 .3 0.5
1 .3 0.5 0 0.5 .1 0.5 .1 0.5 .5 0.5
2 .5 .5 .5 .5 .5 .5 .5 .5 .5 .5
0 .3 .3 .3 .3 .3 .3 .3 .3 .3 .3
1 .1 0 .1 .1 .1 .1 .1 .1 .1 .1
2 0 0.5 0 0.5 0 0.5 0 0.5 0 0

```

Figure B.5: Format of the *.out file, which records the output of the algorithm during its execution.

C:\Documents and Settings\Katie Umberg\My ...Dll Files\Files Used for Testing\output.out 1

/* A sample DLL output (*.out) file. A 1 indicates that the algorithm indicated that a scenario was occurring at that time step, while a 0 indicates that the algorithm felt conditions were normal.

Format:

*Note that (float) Out(R, S, T) stands for the algorithm output for scenario R, sensor S, and timestep T

(int) R stands for the scenario number and is between 0 and NScenarios-1
(int) S stands for the sensor number and is between 0 and NSensors-1
(int) T stands for time step and is between 0 and NSteps-1

```
Out(0,0,0) Out(0,0,1) Out (0,0,2) ... Out (0,0,NSteps-1)
Out(0,1,0) Out(0,1,1) Out (0,1,2) ... Out (0,1,NSteps-1)
Out(0,2,0) Out(0,2,1) Out (0,2,2) ... Out (0,2,NSteps-1)
.
.
.
Out(0,NSensors-1,0) Out(0,NSensors-1,1) ... Out (0,NSensors-1,NSteps-1)
Out(1,0,0) Out(1,0,1) Out (1,0,2) ... Out (1,0,NSteps-1)
Out(1,1,0) Out(1,1,1) Out (1,1,2) ... Out (1,1,NSteps-1)
.
.
.
Out(NScenarios-1,NSensors-1,0)... Out(NScenarios-1,NSensors-1,NSteps-1)
```

The following would be a possible output file if there were 3 scenarios, 2 sensors, and 5 time steps. Note that the algorithm here outputs that an attack is occurring at both sensors at all time steps for the first two scenarios. In fact, the only times that an attack is not signaled here is during the third scenario: for the first time step with the first sensor, and for the last time step with the second sensor.

*/

```
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
0 1 1 1 1
1 1 1 1 0
```


Appendix C

Initial Letter to Developers

This is the initial letter sent to the 29 research teams introducing them to the project, requesting their feedback, and inviting them to participate.

Figure C.1:

University of Cincinnati
Dept. of Civil & Environmental Engineering
Cincinnati, OH 45221-0071
umbergke@email.uc.edu

January 24, 2006

Dear _____:

I am contacting you in regards to the algorithm(s) you have developed for data processing. My name is Katie Umberg and I am a graduate student at the University of Cincinnati in the Department of Computer Science (advisor: Professor Carla Purdy). My thesis work involves quantifying the effectiveness of real-time drinking water Contamination Warning Systems (CWS) and associated event detection algorithms. I am working with the University of Cincinnati Department of Civil and Environmental Engineering (advisor: Professor Jim Uber) and collaborators at the USEPA National Homeland Security Research Center (NHSRC). Throughout my Master's work, I have been working on the problem of designing such a CWS. One important aspect of the problem that has not been closely looked at is the analysis of sensor data received to determine if conditions are normal or if there has been a contamination. I hope to develop an experimental test bed that can be used to evaluate these detection algorithms and then use it to do a thorough analysis of possible algorithms to determine the effectiveness and potential of current methods. I have no bias towards any algorithmic method. Hopefully, this analysis will provide useful feedback for all involved and will move the research in this area forward. The results will inform future evaluations of event detection methods conducted by the USEPA/NHSRC, both as part of the WaterSentinel and TTEP programs.

The main purpose of this letter is to make you aware of this project and invite you to participate by contributing your data analysis algorithm(s) for objective evaluation. I also would sincerely appreciate and solicit your feedback on my preliminary experimental design (attached). Specifically, I would appreciate your thoughts on the following questions:

1. Do you have any suggestions on how I could improve this assessment?
2. Would you be interested in submitting your algorithm to be analyzed?
3. If so, do the specifications I have described seem reasonable?
4. Is there anything that I have not included that would be helpful or would improve your algorithm's performance?

To keep my project on track, I would appreciate hearing from you by January 25, 2006 (see above email address). Based on the feedback I receive, I will write up my final experimental plan, contact those groups who have agreed to participate in this project, and begin the implementation of the plan. If you are interested in participating, please include a short (1 paragraph) summary of your algorithm. I thank you for your time and assistance.

Sincerely,
Katie Umberg

Note: This is not the final experimental design. Changes will be made based on feedback I receive from you and other software developers.

Problem Description

Since the terrorist attacks on September 11, 2001, the United States has been working hard to prevent future assaults and to prepare to respond quickly and efficiently if another strike should occur. Water distribution pipe networks are vulnerable to attack since they are spread out over extremely large areas, have many potential access points throughout the system, and are not currently monitored. Any contaminant injected into such a network could flow directly to countless consumers causing illness or death, as well as large economic costs and severe psychological trauma.

Since it is implausible to protect these enormous networks from contamination, the consensus is that a Contamination Warning System (CWS) is needed to quickly detect if a contamination has occurred so that immediate action can be taken to minimize the damage done. There has been considerable work done in developing algorithms to find the "optimal" placement of on-line sensors. Also, there is continued effort to develop sensors that would be appropriate for these warning systems in their size, sensing capabilities, and durability.

Until recently, however, little attention had been given to the challenge of analyzing the information provided by the sensors. This analysis of sensor data is among the most important steps in developing a successful Contamination Warning System: even perfect sensors placed in ideal locations are useless unless their data can be effectively processed and utilized. Previously, researchers working on the problem of developing a Contamination Warning System largely overlooked the problem of sensor data analysis. They operated under the simplifying assumption that if the concentration of any substance at any sensor is above a specified threshold, contamination has occurred. Otherwise, the network is safe. In this method, detection is seen as a straightforward local decision. In reality, however, the distinction is not so clear.

Perfect, contaminant-specific sensors do not and will never exist. All sensors have some measurement error and are not completely reliable. Also, there is no sensor that can directly detect all possible contaminants. Instead, the current CWS design concept proposes to infer the presence of a foreign substance from analysis of standard water quality measurements such as free chlorine, turbidity, and pH levels (field testing programs to implement such a CWS are underway). These parameters naturally vary in a water distribution system due, in part, to fluctuating demands, pump operations, and treatment plant conditions. Interpreting these imperfect, constantly fluctuating signals is a difficult task. There are no clear-cut rules for determining whether a particular reading is simply normal variation or if a contaminant has been introduced into the system. Even among those in the water distribution system community, there really is not a solid understanding of the fluctuations in water quality parameters. And without a firm grasp on what constitutes "normal" conditions, it will be difficult to accurately distinguish between ordinary and abnormal readings.

Thus the current CWS concept requires analysis of large amounts of imperfect water quality sensor data from throughout the network, and distinguishing between water quality variability due to normal operations, and that due to introduction of a foreign substance. An effective program must minimize missed detections, as damage will continue if a contamination event is not noticed. But false alarms are also detrimental, as response and containment efforts are expensive and the population is

unnecessarily put on alert. High frequency of false alarms would lead to skepticism about the CWS and decrease the likelihood of prompt response for the next alarm.

Project Objectives

The main goal of this work is to develop and implement an experimental computer test bed for evaluating event detection algorithms for use with Contamination Warning Systems placed in water distribution systems. Actual real-world testing is impossible, as real water systems cannot be injected with contaminants and there are currently no networks equipped with a network of optimally placed sensors like those that will be used in a CWS. Also, laboratory testing is insufficient in modeling the changing conditions and complexity of a large distribution system. Therefore, every effort will be taken to maximize the validity and realism under which these algorithms will be tested.

Ideally, this thesis will conclude with a tool for objectively and quantitatively evaluating algorithms that could be used to solve this event detection problem in a CWS, as well as some preliminary data on algorithm performance. The results obtained will likely help researchers improve their algorithms and move closer to an algorithm with the accuracy and reliability needed for a Contamination Warning System. The specific goals of this project are as follows:

1. Do a thorough literature review to understand the current state of the water quality sensor data analysis problem
2. Develop a statistically significant, complete, non-biased analysis tool to compare algorithms for analyzing the data produced by the sensors of a Contamination Warning System
3. Develop the necessary software and gather the needed resources to implement this assessment tool
4. Make contact with as many groups as possible who are working on data analysis and event detection algorithms. Seek to establish what algorithms currently exist.
4. Obtain executable code and algorithm descriptions from at least 2-3 groups
5. Generate realistic simulation data representing both contamination events and non-events for use in testing detection algorithms. These data sets can later be made available to the public and could be used as benchmark data by researchers.
6. Implement the analysis on the obtained algorithms
7. Develop a framework for testing detection algorithms in real-time that can be used by future researchers
8. Give suggestions for future work in this area based on the experience gained from this project

Experimental Design

Data

As mentioned above, contamination detection algorithms cannot be tested in real distribution systems. Therefore, the data for this analysis will be generated by computer using network water quality simulation models, fate and transport models predicting interactions between contaminants and water quality indices, and a Monte-Carlo simulation framework to represent uncertainty and variability in key model parameters and processes. The network simulation model will rely on MSEpanet - a multi-species extension of the widely-used

hydraulic and water quality simulation model EPANET, MSEpanet will be driven by contaminant reaction dynamics models that are specific to the assumed contaminants. The Monte-Carlo simulation environment will generate stochastic simulations of contamination events by considering uncertain or variable parameters such as water usage rates and contamination scenario characteristics.

We believe that this will be the most realistic and extensive data set that has been generated and used for algorithm analysis. Though there are always difficulties with simulation and evaluation, these data will attempt to account for important sources of uncertainty and variability in network hydraulics, as well as relevant reaction mechanisms. EPANET is a popular and trusted water distribution system simulation tool, and it will be used with a network model taken and calibrated from an actual distribution system. Contaminant fate and transport models will be based on existing experimental data for target contaminants. In summary, while challenges exist for developing accurate simulated event data, we believe the data generated will be realistic and will allow objective comparison between different methods.

What I Will Provide To Developers

Each participant will receive a Windows .dll file containing the procedures needed to access the event simulation data, the training data set, a detailed description of the final experimental design, and example code to illustrate use of the included functions.

The training data will consist of 50 separate “scenarios,” each consisting of a different 1-week run on the given network. Each run will be based on different random values of the network hydraulic and attack parameters. There will be *at most* one contamination scenario during each run, and 5 separate time-series representing free chlorine data from 5 different sensor locations will be given for each scenario. Readings will be taken every 15 minutes, so each time series will contain 672 pieces of data (4 readings per hour x 24 hours per day x 7 days). There will be no missing data. All simulations will begin at the same time of day. Information about the contamination scenario (if there is one) during each run will be given for each of the training set simulations.

Sensor data and other necessary event information will be retrieved using the following C-language function delivered in a Windows .dll, which must be linked with your event detection source code (we can provide a Linux shared object library, and it is possible to use this function from other languages such as Visual Basic and Matlab, if needed):

```
float getData(int status, int *timestep, int *scenario, int *sensor, int *truestatus)
```

Arguments:

int *timestep: (output) Identifies the time step of the current data item. Data items are returned in chronological order, and the time step is incremented by each call to getData(). A *timestep value of 0 indicates that a new time series is beginning.

int *scenario: (output) Gives the scenario index (1-50) of the current data item. Note that each scenario will have five data streams associated with it, one for each sensor.

int *sensor: (output) Identifies the sensor index (1-5) associated with the current data time series.

int *truestatus: (output) Identifies whether contaminant is (1) or is not (0) present at the current sensor and time step. This parameter will be disabled during algorithm evaluation (see below), with a return value of -1.

int status: (input) Identifies whether a contaminant alarm was (1) or was not (0) raised (i.e. whether or not the system status was estimated to be abnormal or normal) during the previous time step. The value of status must be set to -1 on the first call to getData(). A valid status value must be provided in order to receive the next data value

Output:

getData() returns a single float which is the value of chlorine concentration for the current sensor index and time step. A -1 will be returned once the function has provided all data from the testing set. A -2 will be returned if the function failed to execute correctly, for example if there was invalid status data inputted.

What I Will Need From Developers

You will need to provide three versions of your algorithm. The three versions should correspond to different target false positive rates: once per week, once per month, and once per 2-month period.

Your programs must be delivered as a Windows executables. Each version of the program (corresponding to the different target false positive rates) may treat each of the five sensors differently. Your program must use the function *getData* as described above to provide the status output for each time step. There will be no time limit for analyzing the data and providing the 0/1 status value, but the total running time for each algorithm will be noted.

Experiment

Once all algorithms have been received, they will be tested using the same test bed software and hardware at the University of Cincinnati. The algorithms will be run to attempt to detect contamination scenarios (if there are any) in a testing set of data that is statistically identical to the training set: each 1-week scenario will be taken from the same network and sensors as the training data, and start at the same time. All algorithms will be tested using the same data set. The status output by each algorithm will be collected and analyzed.

Analysis of Results

There is still an ongoing thought process in regards to algorithm assessment and performance measures. Any input would be appreciated, but the current plan for comparison is described here.

For each scenario, five “status time series” (0’s and 1’s) will be output: one for each sensor. Each of these series will be compared to the correct series. The correct values will be determined by looking at each sensor location for each contamination scenario and identifying when the contaminant was present / detectable at each location. The data series for each scenario will be classified as a false positive, false negative, true positive, or true negative, resulting in 250 total “scores”. The time series for a given event will be classified as a false positive if there is no contamination but the algorithm outputs a 1 at any time, or if the algorithm outputs a 1 prior to the start of the contamination event. A true negative occurs when there is no contamination scenario and the algorithm correctly outputs all 0’s. For a scenario that has a contamination event, a time series will be called a true positive if it outputs a 1 within x time steps of the contaminant arriving at that sensor, where the value of x will be varied as part of a sensitivity analysis. Finally, a scenario will be considered a false negative if the algorithm outputs all 0’s even though there was a contamination scenario or if the algorithm does not output a 1 within x time steps of when the contaminant arrives at the sensor. The totals in each of these categories for each of the three different algorithm settings will be used to estimate the tradeoff between the false positive rate and false negative rate for each algorithm.

Additional metrics will likely be calculated using the binary series. For example, the approximate detection time for each true positive could be found by counting the number of time steps after the contaminant arrives until it is first detected. The number of “false alarms” could be compared by looking at how many 1’s appear outside of the contamination scenario period. Also, the number of contamination scenarios detected by multiple sensors could be calculated, as well as statistics on how many contamination scenarios were detected within n hours. This is a preliminary assessment to explore how well the algorithms can detect a contamination scenario, and it is expected that evaluation procedures will evolve during and subsequent to this study.

Tentative Schedule

January 25, 2006	Receive feedback from developers
February 15, 2006	Mail final experimental plan to developers with details of what needs to be done and all necessary data for completing it
March 24, 2006	Receive final programs from all participating groups
June 5, 2006	Provide all participants with final results

Appendix D

Final Instructions to Developers

This letter will be sent to the developers along with the training data.

Figure D.1:

The testing is finally ready to begin!! I thank you for all your patience in this process. I think I have included all you should need. But feel free to contact me with any questions / concerns / problems. I ask that you send me your final executable(s) by _____.

Files I have included for you:

- Files: `getData.dll`, `getData.h`, and `train.tao`

All 3 of these files should be included in some running directory along with your executable(s). Also, you will need to include the `.h` file in your program and link to the `.dll`.

This will provide you with the *float* `getData(int status, int ×tep, int &sensor, int &truestatus)` function that you will need to access the data and provide your results.

The specifications of the function are described below:

- *status*: This is how you will communicate your results. At every call of the function (except the very first one), you must put either a 0 or 1 here (or receive an error). A 0 indicates that you believe conditions were normal during the previous time step. A 1 signals that your algorithm deduced that conditions were abnormal during the previous time step.
- *timestep*: After each call of the `getData` function, this value will reflect the time step of the current scenario (between 0 and 672 = 4 readings per hour x 24 hours per day x 7 days) of the chlorine value just returned. Note that this will return to 0 when a new scenario is being read.
- *sensor*: After each `getData` function call, this will tell which of the 5 sensors (numbered 0-4) provided the returned chlorine data. Note that the algorithm will return all time steps for one sensor consecutively, and then move onto the next sensor.
- *truestatus*: After the `getData` function is called, you can check this value to see if a contamination has really occurred (1) or not (0). This will give the status of the system during the time step that was just returned. Note that this will be disabled during testing.
- Note that you should call the `getData` function every time you need a new piece of chlorine sensor information. It will step through each scenario sequentially and will cycle through the sensors for each scenario.
 - o Ex:
 1. Time series for sensor 0, scenario 0
 2. Time series for sensor 1, scenario 0
 3. Time series for sensor 2, scenario 0
 4. Time series for sensor 3, scenario 0
 5. Time series for sensor 4, scenario 0
 6. Time series for sensor 0, scenario 1
 - ...
- Your `getData` function calls should look something like this:


```
float current = getData (0, &timestep, &sensor, &truestatus);
```
- The very first time you call this `getData` function, enter a status of -1, so it will be


```
float current = getData (-1, &timestep, &sensor, &truestatus);
```

- * Note that the variables *timestep*, *sensor*, and *truestatus* will not be initialized until this function call
- Once all of the scenario data has been read and provided, the *getData* function will return a -1 to indicate that it is finished.
- If you do not provide a valid status (0 or 1), the function will return -2 and “Invalid status” will be printed out.
- Note that as the program runs, it will create an output file in the directory called *output.out*. You can check this to see all your outputs. This will be used in evaluation.

The Data

50 different scenarios were run using MSEpanet, and the results are given in the included files (you will access these results using the *getData* function). Each scenario represents a one-week simulation, with readings being taken at each sensor location every 15 minutes (672 total time steps). During each one-week period, there was at most one contamination injection, each with a random contaminant type, start time, injection duration, injection strength, and injection location. Note that for a given scenario that contains an injection, each sensor will see the injection differently because of the network hydraulics. Some sensors may never see that contaminant at all. Some may see it for a long period of time. And others may see it more than once if flow patterns change. However, this is irrelevant, since the algorithms will simply analyze the data from each time series (from each individual sensor) and will not be asked to give a network-wide status.

For each time step, the free chlorine concentration will be provided. This is what you will use to infer the state of the system. Also, the actual contaminant status (present or not present) will be given for training. This data comes from the MSEpanet simulations.

Evaluation

I will be writing a simple program (while you are working on training your algorithm) to aid in evaluation each algorithm. It will take as input your algorithm’s output file and a file I will create that contains the true status of each sensor at each time step within each scenario (or the “correct answer file”).

The data will be examined in two different ways: by looking at each time step separately, and by looking at each attack as a whole.

- In the first case, each time step will be classified as either a false positive, false negative, true positive, or true negative. If there was no attack (true status = 0), and you entered a status of 0, that time step will be classified as a true negative. If there was no attack but your status was 1, that classifies as a false positive. If there was contaminant present at the given sensor (true status = 1) and you give a status of 0, that is a false negative. And finally, if there was an attack occurring and you indicate there was an attack (status = 1), you will receive credit for a true positive. Also, you will receive credit for a true positive for the rest of the current attack at that sensor (any response will be considered correct until the true status jumps back down to 0, indicating the attack is over). Note that the sum of the true positives and false negatives will equal the total number of time steps during which an attack was detectable (true status = 1). Also, the number of false positives plus the number of true negatives will give the number of

time steps where the true status was 0. This data (from the different versions of the program) will be used to construct a ROC curve for each algorithm.

- Each attack will also be considered separately. An attack is considered to be a consecutive run of time steps in which true status = 1. Each sensor location will be considered separately, and only time steps in which the contaminant is present at the sensor location are considered to be attack periods (so even though contaminant has been injected into the system, it is probable that the true status at sensor n is 0, because the contaminant is not present at that sensor at that time). Again, note that there may be more than one “attack” in a given scenario due to network hydraulics. In that case, each will be treated as a separate attack to be detected. When considering attacks as a whole, several measures will be considered, including median detection time (where a non-detection has a time of ∞), percent of attacks detected, and percent detected within n time steps.
- Other analyses will likely be performed as well once the data is collected. These may include, but are not limited to, analyses of the different types of attacks (grouped by size, contaminant, ...) and statistical analyses of detection behavior.

What I need from you:

You will need to provide one or more Windows executable files for your algorithm. Ideally, you would provide three executables, each representing a different expected false positive rate so that a ROC curve can be constructed. The goal rates are once per week, once per month, and once per 2-month period.

You will likely have different threshold or learning settings for each sensor, but these should be included in each executable. Again, your algorithm needs to utilize the *getData* function and should be linked to the included .dll. Be sure to clearly include any special instructions for running your program.

FYI: The Network

All contaminant / chlorine data was generated using EPANET and a MATLAB-based randomized scenario generator. An actual water distribution system was used (details of the system will not be given now to ensure that network / spatial information is not exploited for this study), and the 5 sensor locations were chosen by the SPGreedy algorithm developed by Jim Uber at the University of Cincinnati.

The five sites do exhibit different signatures. Included below are graphs showing the water age at each sensor during one simulation. This was chosen to examine because water age is very closely tied to chlorine residuals. Therefore, these graphs also show the basic natural variation at each location.

FYI: The Contaminants

Two contaminants will be used: Sodium Arsenite and Pro-oxonic Organophosphate. These were chosen because of their availability of their reaction dynamics with chlorine, and the fact that they have very different reactions with chlorine. Sodium Arsenite reacts very quickly with chlorine, sending chlorine levels to 0 almost immediately. Pro-oxonic Organophosphate reacts much more slowly, taking several hours to deplete the available chlorine.

I think I have included everything you'll need to complete your part of this project. Again, contact me if you have any questions or need anything clarified. My e-mail address is xxxxxxxxxx and my phone number is xxxxxxxx.

Thank you, and good luck!
-Katie

Appendix E

Sensor Location Attacks

Figure E.1: Locations where attacks were simulated

Figure E.2: Contaminant Presence Caused by Attack 1: 5 hour attack starting 6 hours into simulation, 30,000 gpm injection rate

Figure E.3: Contaminant Presence Caused by Attack 2: 2 hour attack starting 10 hours into simulation, 50,000 gpm injection rate