OptiRTC: A physical computing platform for real-time monitoring and control of environmental systems

A thesis

submitted by

Alexander Bedig

In partial fulfillment of the requirements

for the degree of

Master of Science

in

Civil and Environmental Engineering

TUFTS UNIVERSITY

May 2011

THESIS COMMITTEE

Dr. Richard M. Vogel, Chair and Advisor

Mr. Marcus Quigley, P.E., Geosyntec Consultants\

Dr. Alva Couch, Tufts University

ABSTRACT

A physical computing platform, called OptiRTC, is introduced for augmenting the functionality of mechanized infrastructure with real time remote monitoring (RTRM) and control (RTC) capabilities. In contrast to many single-purpose RTRM and RTC systems, the platform leverages off-the-shelf components and a services-oriented architecture (SOA) common to modern enterprise internet applications to achieve degrees of performance, adaptability, scalability, and affordability that make it suitable for a wide range of applications. Funding for the development of OptiRTC was contributed by Geosyntec Consultants[1] as a means of delivering vertically integrated environmental monitoring and control services to facilitate the adaptive and integrated operation of physical and information systems involved in water management, geotechnical and other environmental control applications. An example application of OptiRTC to reduce combined sewer overflow (CSO) discharge volume at Engine House #3 in Washington, D.C. is discussed.

[1] http://geosyntec.com/UI/Default.aspx?m=MainAboutUs

1.0 INTRODUCTION

Many engineered structures for managing environmental resources are sized to operate under a range of possible conditions, but are not designed to operate optimally over all conditions in that range. For example, a storm sewer system is often designed as static infrastructure for routing runoff away from collection areas in an urban environment. Such systems often operate identically during every storm event, without consideration of the spatial and temporal distributions of the precipitation, or the available capacity of the system to handle the incoming storm. As with many damage-mitigation infrastructures, it is assumed that if a system's loading exceeds its maximum design capacity, then the system will fail and damages will occur. In the case of a storm sewer, such damaging loads frequently arise from storm intensities which were not anticipated when the sewer was designed, or from system modifications that were not anticipated when the sewer was designed, such as urban expansion.

Historical records of environmental events traditionally used to define the expected range of future conditions upon which environmental management infrastructure is designed are becoming less representative due to changes in land use, climate and water infrastructure (Vaughan et al., 2009). Therefore, the ability to adjust existing systems to adapt to conditions outside of the original design specifications is needed. This is particularly true as economic and practical constraints can make rebuilding infrastructure infeasible or unrealistic.

A real time control (RTC) system is defined as a device or set of devices which can manage or regulate the behavior of other devices or systems in real time. An RTC system usually employs a feedback loop, a perpetually repeating set of instructions, operating at a similar time-scale to changes in the system it is designed to control (Schutze, 2004). An RTC system takes measurements of the current system state, and actuates devices which influence the system state. In the case of a storm sewer, an RTC system might be monitoring flow rates in the pipes and actuating valves, in real time, to divert runoff away from pipes with excess volume to prevent overflows. Additionally, integrated RTC systems include information from outside the system, or they may operate multiple and even different systems together. An integrated RTC system that includes a storm sewer may employ predictive weather forecasts into its decision making process, or perhaps it might optimize the release of water from the sewer system to a downstream wastewater treatment plant (WWTP) based on the conditions in both the sewer system and the WWTP (Pleau et al., 2005).

The utility of storm sewers can be improved by the addition of RTC and integrated RTC. The Quebec Urban Community (QUC) project in Quebec, Canada has achieved average volumetric system reductions of sewer overflows of 70% per rainfall event of their entire system. Furthermore, by integrating the release of inline storage with the existing and expected volumes at the WWTP, the QUC is able to use the sewage network to operate the WWTP at its optimal inflow rate (Schutze, 2004). The Northeast Ohio Regional Sewer District in Cleveland, Ohio employs an RTC system that prevents over 700 million gallons of combined wastewater from overflowing into receiving waters each year (Ruggaber, 2007). Generally, these systems are effective because they allow what was once a static, no-control system to respond dynamically and precisely to a random yet predictable loading process, which yields opportunities for more efficient utilization of available system resources, expanding the range of

conditions under which the system can operate effectively and without damage. This paper presents a physical computing platform called OptiRTC that uses off-the-shelf components and a services-oriented architecture (SOA) common to enterprise internet applications to enable it to interface with and integrate a wide variety of environmental management information and infrastructure into centrally controlled, higher-efficiency systems.

1.1 CHALLENGES OF IMPLEMENTATION

The efficacy of RTC systems for storm water and sewer system control as well as many other environmental science applications has been demonstrated (Schutze, 2003), yet many major metropolitan areas in the US and elsewhere still employ static infrastructures for managing storm water and other environmental resources. A review of the existing RTC systems in environmental applications reveals the challenges involved in building and deploying such systems which has curtailed wide scale adoption.

There are two general applications of a real time service: (1) remote monitoring (RTRM), and (2) control (RTC). RTRM systems are designed exclusively for data collection and communication purposes, they do not actuate devices in response to signals they measure. In contrast, RTC systems are designed to operate infrastructure in response to signals either measured by the system itself or transmitted from a remote control source, such as an operator's computer. While there are differences in the purpose, design and components of these systems, they do share many similarities, and thus provide examples of subsets of the total functionality of OptiRTC.

The key components of a traditional RTC system are: (1) control loops, (2) sensors, (3) actuators, (4) controllers, (5) set-points, and (6) data transmission systems (3). Many RTRM systems only consist of sensors, controllers, and data transmission systems, though there are situations in which an RTRM system will have all of the components of an RTC system such as the system implemented by the North Carolina State University Center for Applied Aquatic Ecology (Glasgow et al., 2008).

The most important component of an RTRM and RTC system is the controller. A controller interfaces the components that measure or actuate the physical world with the larger system being monitored or controlled. An RTC or RTRM system is characterized by the parts of it that its controllers interact with. In a centralized system controllers interact with a control room or computer, whereas in a decentralized system controllers interact with other controllers, and possibly with a centralized entity for data archiving and communications. Controllers are typically either analog circuits or Digital Programmable Logic Controllers (PLC). They can be built into or interface with the mechanical components of the system. Components that measure the physical world are known as sensors; sensors produce a digital, analog, or some other form of signal in response to a real world condition. Data transmission systems form the communications layer of the system, facilitating the movement of data between system-specific combinations of controllers, data storage, and external data consumers. By receiving inputs from sensors that are physically or wirelessly connected to the infrastructure of interest, and transmitting those signals via the data transmission system, controllers make measurement information available to other parts of the system. Controllers can also employ logic routines of their own. A control

loop is a sequence of conditional logic statements that run in a continuous loop on the controller, typically at frequencies that are orders of magnitude higher than changes in the phenomenon they observe. Analog controllers perform this function with relay switches, while PLCs use specialized hardware chips or software languages. The conditional logic statements of a control loop compare the values of the sensors the controller is connected to against predefined trigger values, or set-points. When a conditional logic statement evaluates to true, actuators are signaled to fire or change their state, for example, to turn on a pump or open a gate. The technology employed in an actuator is often application-specific, but typically is designed to interact with the circuitry of a mechanized infrastructure, such as the relay board of a pump or gate.

Typical sensors in a RTC urban stormwater system include (1) rain gauges; (2) water level gauges such as floating hydrometers, bubblers, pressure inductive gauges, and sonic gauges; (3) flow gauges such as level-flow converters, ultra-sound velocity meters, and electromagnetic meters; (4) quality gauges for organic pollution (TOC, readily biodegradable COD), nutrients (total, ammonia, and nitrate nitrogen, and phosphorus), biomass (turbidity or respiration activity), toxicity (via respirometry), and others. Operating conditions for sensors in sewer systems and other urban environments pose requirements on sensor hardware, including "measurement accuracy and reliability, physical and chemical resistance, and suitability for continuous recording and transmission" (Schutze, 2004).

Data transmission systems can be based on wired connections including analog wire and telephone networks, or by wireless communication systems, such as radio, cellular systems, or satellite telecommunication devices (Schutze, 2003). The data transmission platform chosen often depends on whether or not the system will use a centralized or decentralized control scheme. While centralized control can be simpler to design, decentralized control offer the potential for improved data transmission reliability, as the system can be designed in most cases such that no single component is critical. In practice, however, both centralized and decentralized systems have been deployed with varying levels of success. Examples of such systems are discussed in the Section 1.2.

While the design objective of an RTC system is for it to be online at all times, in practice the failure of all systems should be considered inevitable. It is essential to plan for any combination of RTC component failures, including data transmission, power, software, and hardware failures. An appropriate design objective, then, is for the infrastructure being controlled to never perform worse than it would in a no-control scenario (Schutze, 2004). Enforcing this objective involves careful consideration of the performance of the existing infrastructure, and may necessitate employing a control scheme that is suboptimal relative to what would be possible if it could be assumed that the RTC system would never fail.

Real time control systems are installed when there are economically beneficial reasons for doing so. Storm sewer overflow, while it can lead to flooding, is not as economically significant as a similar phenomena, combined sewer overflow. A combined sewer is a sewer system that receives inputs from both storm sewers and sanitary sewers. Because of the mixture of nutrients and other contaminants present in the sanitary sewer, a combined sewer overflow (CSO) event can have drastic impacts on both local public health risks and ecosystem health. "Each year in the United States, CSO events result in the

release of 850 billion gallons of untreated wastewater into lakes and rivers, causing drinking water contamination, human illness, animal and fish kills, and eutrophication" (Schutze, 2004). The costs of dealing with these consequences have motivated some municipalities to adopt RTC systems to reduce the frequency and severity of CSO events in their area. The city of South Bend, Florida, for example, estimates the long-term value of the storage its CSO-mitigating RTC system can use to offset CSO events to be ~$3 per gallon (Ruggaber, 2007). The OptiRTC platform can be configured to intelligently manage CSO-mitigation infrastructure, including cisterns, valves, pumps, and alert systems, at the spatial and temporal scales at which CSO events occur.

1.2 INTRODUCTION TO APPLICATIONS OF RTC IN ENVIRONMENTAL ENGINEERING AND SCIENCE

Quebec Urban Community (QUC) has operated a global optimal predictive real time control system that has been in operation in its westerly sewage network since the summer of 1999. The QUC RTC system solves a multi-objective non-linear hydrologic-hydraulic model (SWIFT), in which each linearized objective system is included in a non-linear weighting scheme for multi-objective conflict resolution (Duchesne et al, 2001). The control objectives, in decreasing order of priority, are:

1. Minimization of combined-sewer overflows.
2. Maximization of the use of the treatment plant capacity.
3. Minimization of accumulated storage volumes.
4. Minimization of the variation of set points.

The QUC utilizes a centralized control scheme that integrates flow monitoring, water level, rainfall intensity, rainfall radar, 2-hour rainfall prediction, and WWTP conditions data to operate inflatable in-line storage gates in the main trunk sewer lines. Telecommunications is implemented via a tiered structure in which the central controller communicates with some of the in situ controllers directly, which in turn relay commands to lower-tiered controllers. Telemetry technologies employed in this deployment include 900 MHz radio waves that require an ongoing spectrum license fee, telephone wires, and wide-band radio waves. The central control system solves the optimization problem at 5-minute intervals, providing new set points to the controllers when required. "More than 90%" of the data transmissions are completed successfully and on time (Duchesne et al, 2001). Despite imperfect communications, the system produces CSO volume reductions between 40% and 100% per site, and almost 70% for the system in total (Schutze, 2004).

North Carolina State University Center for Applied Aquatic Ecology (CAAE) has an RTRM system consisting of 18 stations and 4-10 automated platforms in the Neuse River and Estuary, part of the Albemarle Estuarine System. The purpose of the system is to anticipate anoxic events by monitoring for algal blooms, and to provide on-going monitoring of fish populations. It integrates 40 biological, physical, and chemical parameters measured by monitoring stations equipped with off-the-shelf and proprietary sensors into GIS layers that are available in real-time to stakeholders via web-based access (http://www.ncsu.edu/wq/RTRM/index.html). The monitoring stations are placed on platforms fixed to the bottom of the estuary, and use FreeWave wireless modems to send data to on-shore data stations that relay the information to a central server for post-processing and visualization. The CAAE RTRM

system is notable for its RTC-like controller, which is able to actuate a mechanical winch hooked to a YSI 6600ED multi-parameter water quality sensor in order to obtain metrics for the entire water column. After a water-column reading is taken, a DOS-based on-board operating system on the controller performs data filtering routines prior to sending the data to the shore and returning the controller to a low-power sleep mode. The integration of software on the controller allows for adaptability of the system to meet different measurement requirements, as well as a customizable recording interval. The CAAE system currently makes measurements on an hourly timescale (Glasgow, 2004).

2.0 OPTIRTC COMPONENTS AND TECHNOLOGIES

The basic components of OptiRTC are identical to those found in other centralized real time monitoring and control systems; programmable logic controllers interface with sensors and actuators, and a centralized decision-making computer. At regularly spaced intervals, the central computer receives measurements from in situ sensors, runs a set of predefined models with the measured data, and recalibrates the set-points on the controllers with the results. Changes in set points prompt the controllers to actuate connected infrastructure to alter the state of the environmental system's operation.

The fundamental distinction between OptiRTC and existing environmental RTC solutions is that OptiRTC uses Hypertext Transfer Protocol (HTTP) with Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) conventions to interface with its controllers, a design decision that allows for the use of enterprise internet data technologies in the central control component of the system, and expands the possible sources of information useable as time-series data to any piece of information available on the internet. Specifically, this means that any information displayed on an HTML web page or exposed via a web service API is accessible to OptiRTC. Enabling this design are hardware programmable logic controllers that are exposed via a Secure Sockets Layer (SSL) encrypted web service, allowing for credentialed two-way interactions between OptiRTC and the physical-world sensors and actuators.

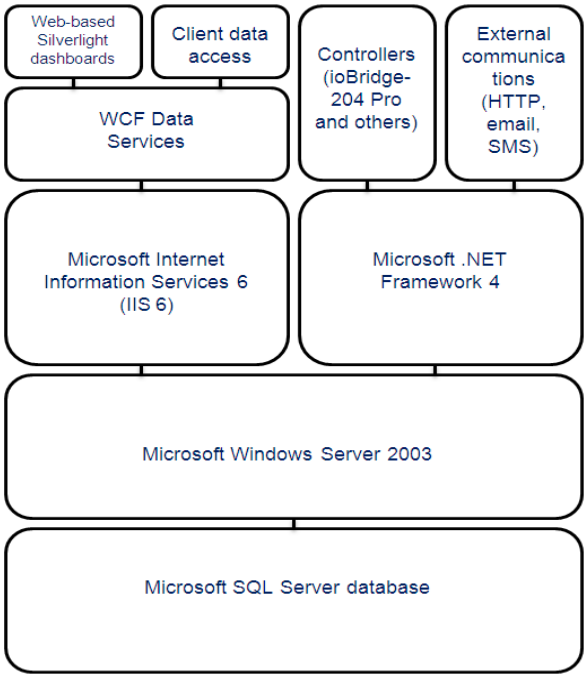


Figure 1: OptiRTC Technology Stack

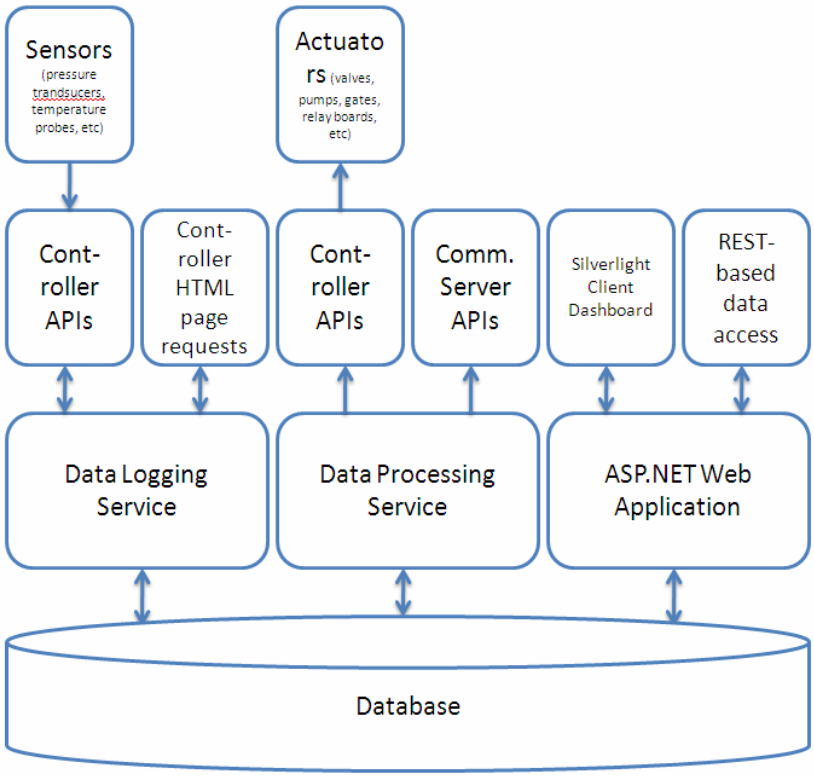
Specifically, OptiRTC is built with a collection of Microsoft internet technologies, including the SQL Server database engine, Windows Server server operating system, Internet Information Services (IIS) web server, .NET Framework software framework and class library, Windows Communication Foundation Data

Services (WCF), and Silverlight. Although controllers can be any internet-accessible information provider or receiver, current deployment plans for OptiRTC specify the use of the ioBridge-204 Pro PLC for physical-world interactions. External communications including HTTP APIs, email, and SMS alerts are handled via their own dedicated servers. The content for these communications is generated by OptiRTC and then passed to the necessary communications server for distribution.

2.1 SERVICE-BASED ARCHITECTURE

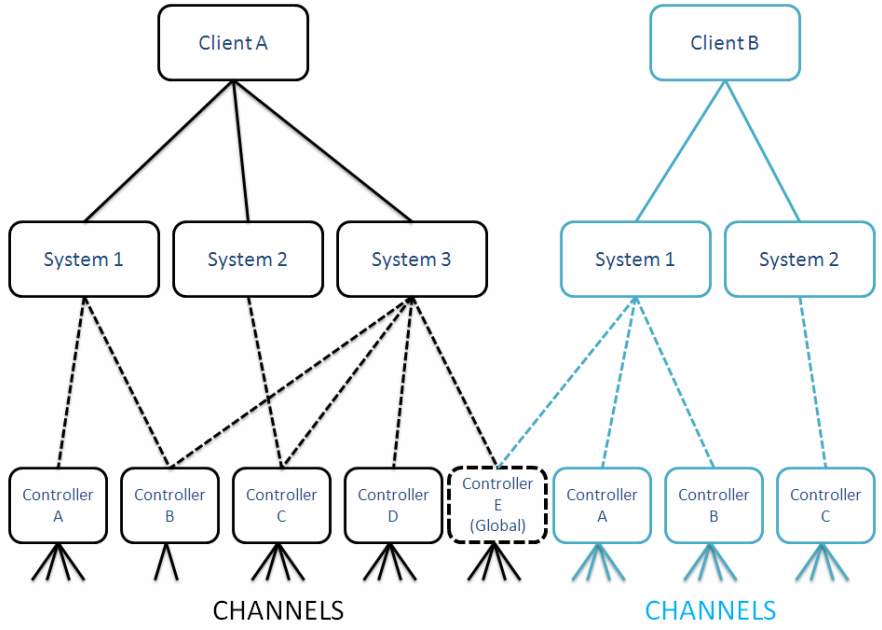
OptiRTC is not a project-specific RTC system. Rather it is a vertically integrated physical computing service designed to support the operations of many unrelated RTC and RTRM projects within the same hardware and software. This extensibility is possible because of the widespread adoption of the internet conventions OptiRTC uses for communications, and the robustness of the database and software framework technologies it is built on. Extensibility was a design goal of OptiRTC because it was developed to be a cost-effective method for adding RTC to a variety of existing environmental engineering services delivered by Geosyntec Consulting, and developing a single system that can be easily parameterized to support independent RTC applications is more cost-effective in the long term than building proprietary solutions for each project.

OptiRTC is a database-driven internet application that involves three independent software services working together: (1) a data logging service, (2) a data processing service, and (3) an ASP.NET web application (Figure 2). The data logging and data processing services are .NET applications written in the C# programming language that run in the background on a Windows Server connected to the database server. In conjunction, these services provide the behind-the-scenes functionality of an RTC system: creating time-series of measurements, running sequences of models parameterized with that data, and sending alerts and actuating devices in the real world according to those measurements and model outputs. The ASP.NET web application, which also runs on a Windows Server connected to the database server, exists to provide controlled external access to data as it moves through OptiRTC. Both the data logging service and the data processing service are versioned and can be set to start and



stop running at specific time coordinates by an administrator of the server they are running on. This combination of features allows for updates to the services to occur without interrupting OptiRTC by installing a new version alongside the existing one, and setting it to start running immediately after the existing version is set to stop.

Information within OptiRTC is catalogued in a hierarchical manner (Figure 3). At the top of this hierarchy is the Client entity. Clients implement identity management functionality in that all other information in OptiRTC is related to a single Client, data processing operations can only use data from controllers for which they have permission, and external data access is limited to information associated with a single Client. Accessing the information associated with a Client requires authentication, which is discussed further in Section 3.1. Beneath a Client are collections of Controller and System entities. Controller entities, as discussed previously, provide one or two-way access to information, sensors, and actuators. Within a Controller entity, each point of information, sensor, and actuator is described as a Channel. System entities are collections of controllers that encapsulate all of the Channels used as parameters in a sequence of model runs. The result set from a sequence of model runs is stored in Decision Space. Each System produces its own independent record of Decision Space data every time its sequence of models is run. Permissions for System access to Controllers are granted for Controllers owned by the same Client as the System, and for Controllers that have been marked as Global (Figure 3, Client A, Controller E). The potential to expand this functionality to allow Controllers to be shared between pairs and groups of specific Clients exists. These constraints are enforced in the database to improve the security of Client data.



The OptiRTC database contains records describing every Controller entity. These records include the date and time data was most recently collected, the desired logging interval, and parameters specifying authentication and other data access details. The data logging service has a base logging interval that defines the base time interval of the system and the maximum logging frequency of any controller, and which is currently set at 30 seconds. Every time this base time interval passes, the data logging service sends asynchronous web requests to the appropriate data access point of every controller for which more time has passed since it was last queried than its desired logging interval specifies. After verifying

that the communication was successful and that the response is valid, the service parses the response and creates a time-stamped record containing every data point this Controller entity collects. As long as the desired logging interval of a Controller is an integer multiple of the base logging interval of the data logging service, this produces a time series of regularly queried responses from the Controller's web service call in the database, known as data logs. By designing the data logging service to parse responses in parallel and as they arrive, it becomes possible to operate many different Controllers at once. A quantitative analysis of OptiRTC performance under simulated loadings is given in Section 3.2.

2.2 CONTROLLER TYPES AND SCALABILITY

Controllers in OptiRTC have different types. Distinguishing each type is the web service it calls to get data, the data access parameters it requires, the method of parsing the results, and the structure of text, temporal, and numeric data that gets stored in the OptiRTC database every time the data logging service takes a snapshot of the Controller's Channels in time. Controller B of Client A in Figure 3 is clearly of a different type than the rest of the Controllers shown, as it has a different number of Channels, and, therefore, at least a different parsing method and possibly a different data structure than the other Controllers. The data structure that corresponds to a Controller type is known as the "space" of that Controller type, and corresponds to a unique table within the OptiRTC database. SQL Server 2005 defines tables, views, stored procedures, user-defined functions, triggers, rules, defaults, and constraints as database objects (Maximum Capacity Specifications for SQL Server 2005). A SQL Server 2005 database supports up to 2,147,483,647 objects, and OptiRTC uses an average of 15 objects per Controller type-space pair. In practice, the quantity of different Controller types that a single installation of OptiRTC can access is much less limited than the quantity of instances of all Controller types active at once, which is limited by the available bandwidth and server hardware. Because every Controller-type uses a different amount of server and bandwidth resources, and each Controller may have a different logging interval, it is impossible to know the real world upper-limit. Future efforts to create independent installations of OptiRTC, run servers dedicated to individual controller types and the special computing demands of parsing their responses, and migrate to a cloud-based server environment should enable sufficient scalability to support very large Systems.

In addition to the parsed data logs created by the system, OptiRTC stores the original, unparsed response from every Controller request in a separate table for archiving purposes. Archival tables, like the data logs, are specific to a Controller type. Archival tables have different indexing and locking properties than the data logs tables, reflecting their purpose as tables that must be quickly written to but never accessed in a time-sensitive manner. In a live implementation, archival tables will be periodically backed up to an external data storage medium, such as a tape system. Although archival data is not parsed, it is possible to replicate any OptiRTC operation that involves its data retroactively. Such calculations would not be expected to operate as quickly because all necessary parsing would have to be executed to extract every value.

The overall size of a database table can affect the time it takes to execute a query on it. Because OptiRTC is designed for real time control, it is necessary that measurements of real-world system state be used in control decisions as quickly as possible. Although a number of options, including careful

database locking and indexing of tables, exist to mitigate the impact of table size on query execution time, there is no solution as effective as keeping the size of the database tables from becoming unnecessarily large. The current system standard is to maintain three months of historical records in the database system, deleting records that are older than this on a weekly basis. This standard could be easily modified for applications that require a longer historical record by creating a distinct Controller type for Controllers of an existing type that need a longer history for live decision-making purposes. Requests for data older than the standard that are not affiliated with OptiRTC decision-making can be handled by an OptiRTC administrator querying the archival tables on a case-by-case basis.

The underlying assumption behind the design decision to limit the length of the historical record available for use in OptiRTC decision-making is that, for most environmental applications, the value of the latest and very recent information is significantly higher than older information. For example, operation of precipitation-loaded systems such as combined sewers requires information on the latest real-world system state, and can rely on external sources like the National Weather Service for predictive metrics that require longer historical records. Similarly, applications that specify calculations to ensure that sensors are not creating unreasonable measurements can rely on a recent historical record to determine a plausible range of values. Furthermore, when historical values outside of the three-month window are required for a decision-making calculation, it is often actually the stochastic properties of the historical time-series that are relevant to the decision being made, not each individual value. Because all old information in OptiRTC was at one time new information, an ongoing calculation of these stochastic properties is possible, and will always be available.

2.3 SYSTEMS AND MODEL SEQUENCES

An OptiRTC System exists to define a sequence of models and a set of rules describing when this sequence gets run. A model is broadly defined as a function that takes a result set of one or more queries of permissible data as an input. Because all models in OptiRTC are written in the .NET Framework, these functions can range from simple data manipulations to the execution of any Windows-based program that exposes an API or is controllable via Windows Component Object Model Automation (COM). Models use input data to either create a decision space variable, or send an external communication, but not both. As a sequence of models is run, decision space variables are added to an in-memory instance of a decision space record. Models that run later in the sequence are able to use data from this in-memory instance. When all of the models in a sequence have been executed, and were executed successfully, the completed record of decision space data is recorded to the database. This design allows for decision space data that is communicated to the outside world to be created by one model and then sent by another without having to access the database for the original input data more than once. It also facilitates the immediate use of model results for communications, temporally prioritizing real-world control over data archiving. Permissible data includes values from the data logs of any of the Controllers a System has access to, values from models in the same sequence that ran previously, and information from two System-level entities known as System Constants and System Static Resources.

In contrast to Controller-based data, System Constants and System Static Resources are not time series, and are never created, updated, or deleted except by an administrator. System Constants are named and ordered sets of numerical values assigned to a single System by an OptiRTC administrator. They are used both to provide constant values involved in a numerical calculation run in a model, and, where multiple System Constants have the same name and incrementing order, lists of data suitable for combining with other lists of System Constants to create a lookup table.

System Static Resources are similar to System Constants in every way except that they represent text-based data instead of numeric. Primary uses for System Static Resources involve pre-scripting communications templates that a model can fill in with Controller-created or decision space data, and facilitating the real-time operation of software packages that require structured inputs. For example, the United States Environmental Protection Agency publishes an open-source software package called the Storm Water Management Model (SWMM) that facilitates the calculation of a number of metrics relevant to urban water management, including the in-sewer volume response to a given rainfall event (Storm Water Management Model User's Manual Version 5.0). SWMM uses specially structured input data files to run, which can be stored as System Static Resources, combined with real-time metrics by OptiRTC models, and then used to make real-time determinations of the urban watershed state. A live implementation of this functionality has not been created, but is expected in the first half of 2011.

A central mechanism of OptiRTC is the concept of data mapping. By naming the data space, Controller within that space, specific data object, and a time period of interest for non-constant data spaces, a single data mapping entity uniquely describes any piece of information in OptiRTC. Collections of data mappings define the inputs of every model in a System modeling sequence. Access to information by a given model is restricted by only permitting data mappings that include combinations of data space and Controllers to those the System of the model has access permissions to.

At the most basic level, System model sequences provide an interface for using real-time information in almost any computation process. In practice, however, the kinds of computation processes that can be used in model sequences are limited by the duration of time it takes to complete them as compared to the frequency of the System model sequences and the period of time during which the results can still be valuable for decision-making. These values are highly application specific, and their determination involves the consideration of existing best management practices, tradeoffs between precision and decision-making speed, and the client's willingness to pay for dedicated server resources to support the calculations. A more detailed discussion of the rules surrounding System decision making is included in Section 3.0.

2.4 THE IOBRIDGE-204 PRO PLC

The ioBridge-204 Pro programmable logic controller and web service facilitates reliable internet-based communications with a wide range of real-world sensors and actuators at time-scales in the hundreds of milliseconds. Crucially, it provides this functionality at a price two or three orders of magnitude less than traditional SCATA PLCs, which range between the tens and hundreds of thousands of dollars (Ruggaber, 2007). The ioBridge-204 Pro PLC (Figure 4) requires 100-240V~0.7A of AC power at 47-63Hz, and a broadband internet connection provided by a CAT-5 Ethernet networking cable. It provides four Channels of data that interface with sensors via standard 5-pin connectors. The connectors can interpret sensor responses as 10-bit analog-to-digital converters (resolution of 0-1023), and native digital input and output, and send serial strings as output. In addition, the PLC provides one two-way serial port, and expansion modules for the 5-pin connectors that enable relay control, servo control, and X10 power system interfacing. These options are sufficiently expansive to assume that most measurement devices with an electric circuit could be read with an ioBridge-204 Pro PLC.

For protection from the environment, the PLC is contained in a NEMA rated enclosure that is both waterproof and capable of supporting an operational temperature range of -40 to 185 degrees Fahrenheit (Figure 5).

n ioBridge-204 PLC

The ioBridge-204 Pro PLC has sufficient on-board memory and firmware to enable it to perform a number of on-board logic rules, programmed via an internet GUI-based ladder logic-style programming language. These rules can be set to respond to thresholds or changes in the measurements made by the Channels, or to act on a periodic basis. While there is enough memory to accommodate many logic statements, there is an upper limit on the number of commands that can run on-board. In addition, each Channel on the PLC contains two 16-bit memory spaces and one 8-bit memory space. The rules can set and respond to the values in each of these memory spaces. Notably, there is no inherent capacity for local storage of historical values, although it is possible that the 2-way serial interface could be used to implement a proprietary solution. This limitation means on-board logic can only make use of the latest set of variables.

The PLC is tightly integrated with a cloud-based web service. Every 10 seconds, the PLC syncs its measurements with this cloud service, which then distributes the data to properly-credentialed web requests such as those made by the OptiRTC data logging service. Additionally, the web service allows for Controllers under the same user account to share measurements, set the values of many of the on-board variables via a mathematical expression with the data from any of the Channels, and use the National Weather Service's Probability of Precipitation (POP) and Quantity Precipitation Forecast (QPF) feeds as variables.

Importantly, both the PLC and web service are network-state aware, meaning the data feed accessed by OptiRTC will know when a PLC has not updated its values with the latest information, and the PLC itself will be able to implement a different set of on-board logic.

The combination of these features produces a standard control interaction with OptiRTC that mirrors set-point based control systems common with centralized control schemes in RTC. OptiRTC is able to control target values that on-board logic can determine how to pursue, switch between different sets of on-board logic, and respond appropriately when a networking failure occurs.

The networking stack of the PLC is a proprietary technology developed by ioBridge, and it is robust. It works by analyzing local network traffic for available ports to the outside internet and attempts to mimic what it observes. IoBridge claims that if a PC can connect to its website via an Ethernet connection, then an ioBridge PLC would function on the same connection. One of the crucial features of the networking stack is its boot to on feature, which enables the possibility for using external relays to cycle power to the router, other networking devices, and even the PLC itself in the case of a networking failure.

By itself, the ioBridge-204 Pro PLC is a capable device for facilitating RTC and RTRM applications. However, despite the flexibility of its available server-side and on-board logic, its limitations involving the use of historical information in decision making and the complexity of its expressions make it insufficient as a standalone solution for advanced decision making systems.

3.0 THE OPEN-WORLD PROBLEM

An environmental RTC system implements a consistent set of algorithms to determine the proper methods for controlling infrastructure. Because the environment it is regulating is a continuously dynamic system, the frequency of and rules governing the initiation of a decision-making sequence are major factors in the performance of an RTC system. Importantly, while an environmental system is dynamic, the measurable parts of it do not necessarily change on the same time scales. For example, the temperature of water in the first foot of depth of a reservoir will always lag both the intensity of the sunlight, and the running monthly average air temperature, and at different rates. Moreover, existing regulations and conventions within environmental engineering practice can specify a certain sampling frequency that is sufficient for the control of systems addressing particular problems. Because higher sampling frequencies are more costly to operate (see Section 3.3), there is little incentive to monitor or use those values at a higher frequency than the convention stipulates. Further, there is no purpose in running a decision-making sequence with the same set of input data twice, and in fact doing so could

result in the improper weighting of measurements in systems where the historical decision trend is an input variable to the next decision. Therefore, since Controllers in OptiRTC set their own data acquisition rates, the decision rate of a System must be equal to an integer multiple of one of these Controller sampling frequencies. By default, OptiRTC assumes that it will not initialize a new decision-making sequence unless all Controllers have acquired new data since the last decision was made. In cases where the System decision rate is proportional to a Controller with a higher sampling frequency than the lowest frequency Controller in the System, the System must be set to be able to use non-new data from the Controllers with a lower sampling frequency than the System decision frequency. Furthermore, this mechanism is applied to provide flexible coping mechanisms for a problem computer scientists call the "open-world" problem.

A well-recognized challenge in physical computing and network applications is the open-world problem: data will be sporadically and unpredictably delayed or otherwise unavailable, transforming the previously deductive problem of decision algorithm design into an abductive problem where the optimal problem solution cannot be preconceived (Faltings, 1995). In the OptiRTC case, failure-induced abductive problem space is manifested in five scenarios. The behaviors of the overall system in each of these cases is dictated by the principle that all outcomes must be as good or better than the best no-control outcome (Schutze et al., 2004), which are adopted on a case-by case basis from the specifications, conventions, and common practices of the environmental engineering field. OptiRTC employs pre-determined, "good enough" deductive solutions to each of the failures that cause open-world decision conditions.

First, there will be times when some or all of the Controllers of a System do not log data successfully, or are delayed such that the next query of the Controller is initiated before the previous query returns. In this case, it must be noted that the PLC is still online and capable of receiving commands, but something between the Controller web service and the database failed, either in the data logging service or in transmission. Section 3.2 covers these events in more detail. In this case, an application-specific decision is made when the System is configured regarding whether or not data from a Controller can be used in a decision space calculation when it is not as new, or as "fresh" as it ideally could be. Awareness of this state is facilitated by an internal data versioning system that tracks the update state of every Controller from the perspective of every System that uses its information. The decision to allow non-fresh data use must be made with an understanding of the implications of running models with shorter time series for the Controllers that failed. Models that expect data but receive none do not successfully execute, triggering the third condition discussed below. When using non-fresh data for a Controller in a specific System's model sequence is allowed, an expiration period is also specified describing a time period during which this non-fresh data stream is still useable. One of the on-board memory slots of a PLC is allocated to being a state variable; the combination of this value and whether or not the Controller detects that its network connection is active determines the set of on-board logic that governs its behavior. The primary purpose of the state variable is to give the Controller a means of understanding that, despite its active network connection, a new decision update is not coming down from the System. An on-board periodic rule can be set proportional to the expected System decision making frequency that decrements this value until it reaches zero, at which point a different logic set is employed. When a

System successfully sends updated decision data to a Controller, this value is reset. Because the PLC is aware of its network connection during this period, it would not switch over to network failure logic preemptively.

Second, there will be times when OptiRTC model sequences do not evaluate before the same model sequence is initiated again. This scenario can occur during loading spikes on the database server or web server infrastructure. In this circumstance, the delayed decision calculation is terminated in favor of the newer one. Because all System decisions are processed in parallel, it is possible that the newer calculation could be overridden by a decision employing less-current information if this behavior were not adopted. The disadvantage of this approach is that if a value in the cancelled decision space would have triggered an external communication like an email, but the next measurement would not, the communication may never be sent. It is expected that OptiRTC will be able to support sufficiently fast Controller sampling frequencies that this should not be an issue. Even if it were to arise, the historical record should still include the alert-worthy value, which may trigger other kinds of communications by other Systems. While this solution is imperfect, its potential damages should be minimized by ensuring that the Controller sampling frequency exceeds that of significant changes observed in the real world system. Moreover, development effort has not been invested in determining an efficient method for determining if two decision-making sequences, running on separate processing threads, are of the same System in order to facilitate a different behavior.

Third, there will be complications with the computing systems of OptiRTC or ioBridge, and decision information will either not be created or communicated to the PLCs. This case is handled identically to the second, with the PLC running through its periodic backout sequence to determine if it should switch to a different set of on-board logic. Because OptiRTC will read the Controller state again in the next iteration, it will observe that the Controller did not make the expected adjustments, which will likely result in decisions similar to those that were not successfully communicated.

Fourth, a network failure completely isolates a controller from the outside world. This circumstance is different than the first scenario because the Controller is aware that its network connection is down. In this case, the network-state aware PLC switches to a separate set of on-board logic rules that are designed to both protect the resources and devices it manages, and mitigate the risk that many systems experiencing this kind of failure in close spatial or temporal proximity do not cause some unnecessary harm via collective action. Ruggaber et al. describe in a case study implementation of CSONet in South Bend, Indiana that a constant, continuous release of water after a storm mitigates the risk of an RTC system-induced CSO event (Ruggaber et al., 2007). Generically, the principle of gradual system change as a means of mitigating the risk of system shocks and the associated undesirable outcomes is intuitive. Because both global predictive control and historical trend-based decision making require network connectivity with the ioBridge-204 Pro PLC, most solutions expected to be implemented with OptiRTC will employ offline logic that makes less aggressive decisions than an online system in order to behave as much in accordance with standard operations while minimizing the risk of RTC-induced negative outcomes. Where this is not possible, network failures will be treated as complete power failure scenarios.

Last, an electrical outage may cause the PLC and possibly one or more electrically-actuated devices to lose power. In cases of complete power loss, the solution involves careful infrastructure engineering that is designed to fail in a benign manner. Effectively, a complete power-loss scenario must be handled as a transformation of the active infrastructure into its passive corollary. A gate that selectively isolates a saltwater marsh from the ocean, for example, would be set to fail into the open position so as to not interfere with tidal actions. Similarly, a storage resource that is operated for CSO mitigation during online events would be implemented with traditional static overflow capacity to allow it to function without a predictive control.

In all cases of system failure, the appropriate solution is going to be inevitably application specific. Varying factors including other components of the system, regulation compliance, and the priorities of the owner make for an infinite number of possible desirable system failing behaviors. OptiRTC provides state-aware mechanisms in the central control scheme, the PLC, and in the management infrastructure for specifying system behavior under a range of failure scenarios. It is possible that being able to provide integrated designs of both the infrastructure and monitoring and control components of a project will enable Geosyntec to deliver more RTC systems than organizations that focus on a single component.

3.1 SILVERLIGHT DASHBOARDS AND CLIENT WEB SERVICES

It is not sufficient for RTC systems to simply measure and control, they also require user-friendly operator interfaces (user interfaces) to display and summarize the current system status to clients and administrators (Schutze, 2004). The internet technology behind OptiRTC transforms providing secured access to information into a simple task of building a web application. The .NET Framework and Windows IIS web server make this task straightforward by providing widely-adopted options for web application development that tightly integrate with the SQL Server database. OptiRTC employs a WCF layer running under an ASP.NET web application and IIS server that provides access to data conditional on the authentication of the user. Authentication is tied to the Client object in OptiRTC, managed via the settings of the ASP.NET application, and enforced by the IIS server. Tiered levels of access privileges, known as roles, enable public, private, and administrative levels of data access.

The WCF layer exposes web services that support data access via the REST convention of parameterized Universal Resource Indicator (URI) web requests. These services divide OptiRTC into three internet-accessible components: (1) authentication, (2) system structure, and (3) real-time data. The authentication service handles details related to verifying a user's identity, the system structure service describes the collection of information an OptiRTC administrator has made available to the client, and the real-time data service provides access to Controller-type and Decision Space data. Both the system structure and real-time data services require successful authentication before they will return data assigned to a particular Client, although a demonstration System is publicly accessible. Importantly, the system structure service implements the same data mapping conventions employed in the decision making model sequences of OptiRTC. In this manner, information from the real-time service can be queried for and retrieved with the complete metadata an OptiRTC administrator assigned to it. Viewed in this context, OptiRTC web services function as a content distribution system, allowing Geosyntec

engineers to create metadata-enriched feeds of real-time measurement and model-result information that are accessible via the internet.

The WCF web services are consumable in two ways: (1) via an Open Data Protocol-compliant interface, and (2) in a Silverlight web application. The Open Data Protocol provides a web standards-based mechanism for clients to use third-party applications to access data for those Client accounts they have access to. For example, a scientist may use this feature to query live data from an Excel spreadsheet to perform additional analysis, or a programmer could use it to build custom web pages with the information. By providing access via a standards-based internet interface, OptiRTC is able to integrate with a wide variety of uses for its data.

Rather than rely on the efforts of clients to build web sites, spreadsheets, and other methods for exploring their information, OptiRTC provides a cross-platform dashboard application for visualizing and interacting with its data. Based on Microsoft Silverlight technology, a browser plug-in similar to Adobe Flash, the OptiRTC Dashboard Client application integrates the three WCF services into an easy-to-use, automatically updating tool for exploring and visualizing the real-time information in OptiRTC. The dashboard can be seen in the browsers running on both Windows and Mac OS personal computers; Linux is possible although presently untested, and mobile platform support is currently unaddressed. Although development on the Dashboard Client will be an iterative, on-going process, it is currently live at http://optistorm.geosyntec.com (soon to be https://optirtc.geosyntec.com, click on Real-Time Dashboard to see the dashboard). Its existing capability allows for any time-series described by a data mapping to be visualized as data series on over 20 different kinds of charts. The Dashboard Client refreshes its information every 30 seconds without reloading the page, ensuring that the user always sees an up-to-date picture. A System can have an unlimited number of dashboards assigned to it, each of which consists of a layout of charts and data mappings. The Dashboard Client interprets the metadata information provided by OptiRTC to create a visual context for the information it presents. A screenshot of the public demonstration System dashboard is shown in Figure 6.

In addition to providing data display, the Dashboard Client is capable of performing transformations of client-side data. For example, all data in OptiRTC is time-stamped with UTC time coordinates to protect against incorrect decision-making during daylight savings events. The Dashboard Client transforms this timestamp into local time coordinates for visual display by using the UTC offset value specified by the clock of the computer running it. Future plans to expand this functionality will allow the user to transform live data within the Dashboard Client to facilitate real-time exploratory data analysis.

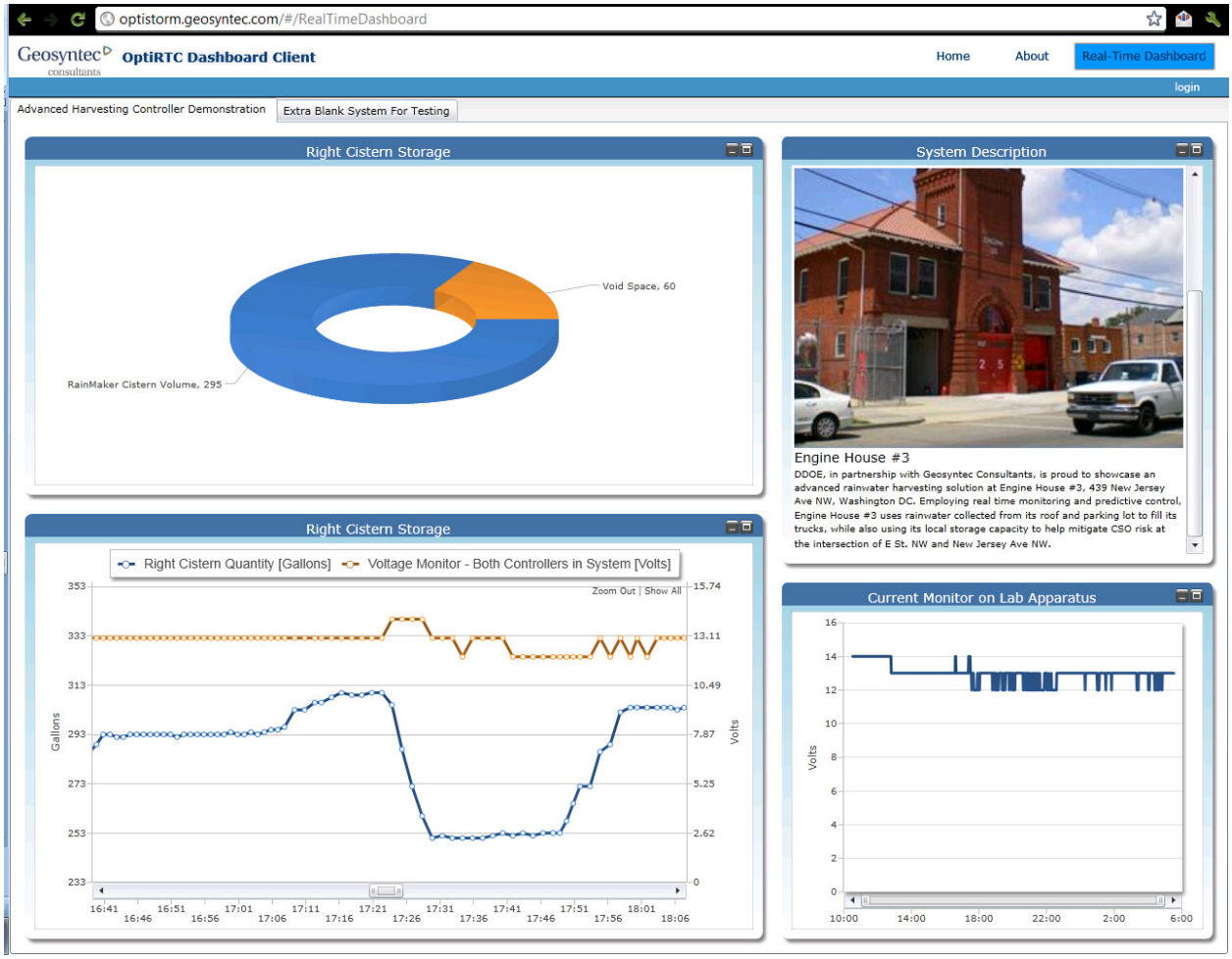


Figure 6: Screenshot of OptiRTC Dashboard Client showing the public System demonstration dashboard

3.2 STRESS-TESTING OPTIRTC

The performance of OptiRTC is highly dependent on a number of factors, including the availability of database server and application server memory and central processing unit (CPU) cycles, application server bandwidth to the internet, geographic proximity of the server facility to Controller service providers, and overall internet congestion. At the time of this writing, OptiRTC has been deployed to a shared SQL Server database server and a VM of a Windows Server in a collocation facility operated by Geosyntec Consultants. The database server supports 38 other databases, including two public-facing databases that may support hundreds of simultaneous connections, and is managed by an IT staff that has not been instructed to take any special precautions when administering or backing up the OptiRTC database. It runs on an NT AMD64 platform, and has two processors and 22526 MB of memory. The VM Server, where the OptiRTC services and ASP.NET web application run, has an Intel Xeon 5130 CPU clocked at 2.00 GHz, and 512 MB of memory. Because the hardware supporting OptiRTC is not dedicated to that task, the following stress tests were conducted to demonstrate sufficient scalability and performance to support several projects on the scale of the District Department of the Environment

(DDOE) Engine House #3 (EH3) project, the first real-world application of OptiRTC. Details of this project are discussed later in this paper.

A primary task of models in OptiRTC involves translating an analog representation of a measurement into a value that is useful for making decisions. A common example of this in water management applications is that of a storage resource. Measuring the volume of water present over time is a common requirement for an effort to manage the storage. Water volume in a storage resource can be measured indirectly by using a pressure transducer to get a reading of the height of the water in the storage resource. This height can be transformed into a volume with a storage-elevation curve, which, for cases like a storage resource shaped like a cylinder and positioned with its ends perpendicular to the ground, is not necessarily linear. This transformation is done with a lookup table as described in 2.3 on Systems and Model Sequences. Other common tasks include calculating running historical statistics, comparing model results to pre-defined action-causing thresholds, creating and sending alerts to email and short message service (SMS) messaging services, and sending values to PLCs for real-world actuation.

One of the advantages of internet controllers is that the physical networking connection is already standardized everywhere in the world. This means that, unlike many RTC and RTRM systems that incorporate proprietary communications technologies, it is simple to deploy a test environment of OptiRTC in an office. Furthermore, it is reasonable to expect that geographic location would not noticeably change the performance of an internet-connected Controller at the tens-of-seconds timescale relevant to OptiRTC, because the vast majority of communications across the internet tend to be resolved on timescales measured in fractions of seconds.

Figure 7: Rainfall Simulator and Demo Cistern

A demonstration cistern was created to simulate the impact of rainfall events on a storage resource. A Poseidon PS3 external aquarium pump was plugged into a power supply actuated by a power relay board connected to an ioBridge PLC, which was put into a System in OptiRTC. This system was assigned a Weatherbug API-type Controller, set to query for the running total of the current day's rainfall every 5 minutes at the geographic coordinates specified in the Controller configuration. By using a model sequence that converted the response to an elevation, OptiRTC is able to pulse the pump for a duration of time such that it pumps enough water from a bucket into a PVC water column to emulate the elevation rise that would be observed on a tank with 1/10th the cross-sectional area of the area of its drainage

basin (Figure 7). In this manner, the impact of any live rainfall event in the world can be simulated on the storage resource for the purpose of testing OptiRTC. It should be noted that the Weatherbug API-type Controller is not part of a commercial product.

Based on expected tasks for the EH3 project, an experimental set of ioBridge PLCs were set up to monitor the storage resource. These controllers were given global access permission, and additional Clients, Systems, and Controllers were created to use their output. In total, there were 10 active Controllers being used in 15 active Systems. Each of these Controllers were set to use a 60-second sampling frequency, and Systems, which included two Controllers each, were set to allow 6-minute expiration periods for each Controller to allow for processing despite any short-lived network failures. These Systems all ran the following model sequences:

1. Convert elevation measurements from the cistern 8-bit analog-to-digital converted measurement to gallons, employing a lookup table. This used an exaggerated scale to produce demonstration dashboards that appeared reasonable in scale for a real-world deployment.
2. Calculate the arithmetic mean of the gallons and analog scale number.
3. Calculate the geometric mean of the gallons and analog scale number.
4. Calculate the running 7-day total of the elevations of the cistern.
5. Create an XML representation of all Decision data created by models 1-4.
6. Calculate the void space between the existing volume and the known, constant maximum capacity of the system.
7. Set a Variable value on-board one of the Controllers to the value of the void space.

Running these Controllers and Systems simultaneously was designed to be a sufficient stress test to determine if OptiRTC would be capable of operating the Engine House #3 project and others of its scale.

Figure 8 shows the difference between the time the data logging service initialized a Controller-querying event and the time each successfully parsed Controller request was committed to the OptiRTC database, and Tables 1 and 2 summarize this information.

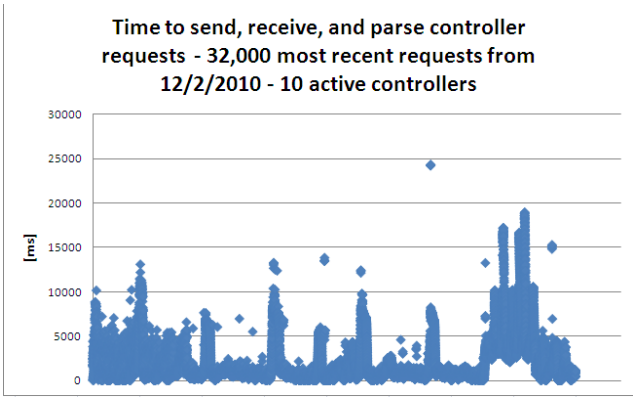


Figure 8: Data logging service performance during OptiRTC stress test

Table 1: Data logging service performance statistical summary – 32225 total requests

Statistic	Time to request and process a single data point [ms]
Average	1961.971
Median	1010
Standard Deviation	2367.992

Table 2: Data logging service performance summary – 32225 total requests

Data logging completion threshold	Count of experimental requests that completed within the threshold	% of total experimental requests that completed within the threshold
Less than 2 seconds	23041	71.50%
Less than 3 seconds	25277	78.44%
Less than 5 seconds	28897	89.67%
Less than 10 seconds	31828	98.77%

Table 3: Decision making service performance statistical summary – 32002 total requests

Statistical summary	Time to complete a full decision space cycle [ms]
Average	9497.603
Median	9730
Standard Deviation	3452.674

Table 4: Decision making service performance summary – 32002 total requests

Data processing completion threshold	Count of experimental requests that completed within the threshold	% of total experimental requests that completed within the threshold
Less than 10 seconds	17430	0.544653
Less than 15 seconds	31050	0.970252
Less than 20 seconds	31634	0.988501
Less than 40 seconds	31987	0.999531

Figure 9 and Tables 3 and 4 convey the same information for the accompanying model sequences that were run during the stress testing period. The results of the OptiRTC stress testing suggest that, despite lacking dedicated hardware, OptiRTC is probably sufficiently robust to support a variety of control scenarios. It is worth noting

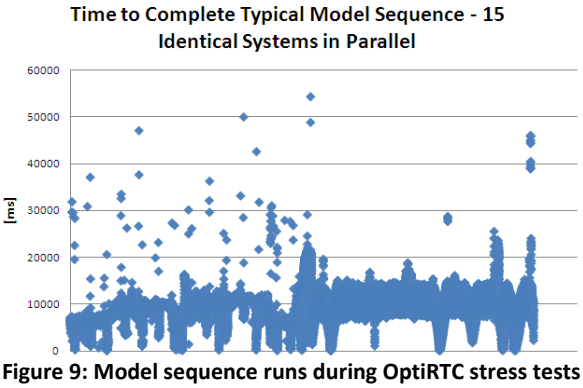


Figure 9: Model sequence runs during OptiRTC stress tests

that the system does slow down periodically between 7:00 and 8:00 UTC (times not shown on the chart axis). While those aren't the only times the system slows down, it is the most consistently slow period. It is currently believed this occurs because of conflicts with the regularly scheduled database backup and maintenance procedures that are conducted during that timeframe. Migrating to a dedicated server resource and developing backup plans that are sensitive to the application of

Communications during the sampling period were 100% successful, but 16 days is not sufficiently long to capture the picture here. The stress test has been operational since late September 28, 2010, with periodic periods of downtime for further development. Adding in the rest of the data, over 99.69% of the possible communications went through successfully. Of the 308 that failed, 51 were caused by OptiRTC database connection timeouts typical of an overloaded database server, and the rest involved Offline responses received from the ioBridge web service.

It is not just adequate that the communications between OptiRTC and its Controllers complete successfully. They must complete successfully within a time-frame that provides a reliable basis for real-time control. In line with the 10-second maximum temporal resolution of the PLC sensor data, the critical performance statistic of the data logging service is the success rate for acquiring data within 10-seconds of initiating the web request. During stress testing, OptiRTC met this statistic over 98.7% of the time. Similarly, the performance of the decision making service suggests that OptiRTC is ready to support Systems with decision making frequencies of 60 seconds. Stress test results indicate that over 99.9% of the decision model sequences completed in less than 40 seconds. Assuming a delay of 5 seconds between the completion of a data logging cycle and the initiation of a decision making cycle, these statistics suggest that (0.9877 * 0.9995 = 0.9872) OptiRTC will make decisions in less than 60 seconds with a success rate of at least 98.72%. Note that this statistic is actually reflective of a 55-second complete cycle of OptiRTC, and is, therefore, conservative.

The performance of the current implementation suggests that OptiRTC is capable of facilitating RTC and RTRM operations at a higher temporal resolution than many existing CSO systems, including the 15-minute frequency of CSONet (Ruggaber et al., 2007), and 5-minute frequency of the QUC CSO system (Schutze, 2004). To the extent that statistics discussing system reliability are available, OptiRTC performs comparably and favorably to these systems as well, although to be fair, OptiRTC has not yet been proven in a real-world setting.

3.3 OPTIRTC RESOURCE CONSUMPTION AND OPERATING COSTS

Data storage space, server CPU loads, and external internet bandwidth are the major resources consumed by OptiRTC on an ongoing basis. The resources consumed by a given OptiRTC System, however, are highly variable. Factors that influence resource consumption include the number, type, and sampling frequency of Controllers, the model sequences being run, and expectations for the number of users who will be simultaneously viewing online dashboards and accessing data via web services. Furthermore, due to its existing operating status on shared database and web servers, precise estimates of CPU loads and bandwidth consumption are not available. Still, based on the records created during development and stress testing that were heavily dependent on the ioBridge-204 PLC

Controller-type, estimates of the data storage consumption are available. A single OptiRTC ioBridge-type Controller record costs 0.4166 KB, which scales to almost 18.6 MB for a 31-day month at a data logging frequency of one log per minute. Similarly, a single Decision Space record produced by the action sequence used in the stress testing costs 2.6498 KB, which scales to 118.3 MB for the same period assuming perfect operation and decision making routines scheduled for every update of the controller. For the systems in the stress test, which have two Controllers and produce Decision Space records each time both are logged successfully, the storage space consumption rate is 155.5 MB per month. Assuming that these systems maintain a 3-month historical record with up to 7 days of overflow between deletion and archival operations, the storage requirements for a single System with a 1-minute operational frequency is 501.6 MB.

While these computing resources are currently supported by the excess capacity of systems that were purchased for other projects, it is possible to roughly estimate operating costs in 2010 by comparing that resource consumption with the pricing of hosting plans that could support such operations. Because its technology is based on the same SQL Server database engine, and its server packages could easily support the processing requirements of OptiRTC, Microsoft Azure Cloud Services is a reasonable choice for this comparison. At the time of this writing, 10 GB of relational storage and a low-power server that still exceeds the resources currently allocated to OptiRTC costs $109.95 per month. Making the reasonable assumption for small-scale projects that this package includes sufficient bandwidth to support OptiRTC operations, this resource could support 19 Systems at a cost of $5.79 per system per month. Adding on fixed costs of $5 per month per ioBridge-204 Pro Controller, and the total ongoing operation cost of each System is $15.79 per month – just under $2 per time series per month. In reality, OptiRTC systems are likely to vary widely in complexity and cost, with the largest unknown being the operational costs of Systems that are accessed regularly by large numbers of users. It is worth noting that it is reasonable to expect the operational cost of all internet computing resources to continue to decrease over the long-term.

3.4 DATA PROVENANCE

The most efficient information model for tracking the metadata of measurements involves describing that information by the attributes of the process that measured it (Lehning, M., et al., 2009). OptiRTC currently employs a partial implementation of the data provenance concept, largely because of the possible layers of abstraction between OptiRTC and the original source of its data, which themselves may be changing without necessarily exposing that change to OptiRTC. For example, while the metadata associated with a measurement of temperature by a specific model of temperature probe may be simple and constant, the metadata associated with a measurement retrieved from a web site or web service may be incomplete or dynamic. This condition commonly arises in cases where the data being ingested as a measurement is the output of a complex model itself, such as a predictive weather forecast. Although the metadata of the values provided by predictive weather forecasts includes many details (see Appendix A), and a separate service exists to automatically provide notification of changes to this metadata (NWS XML Services Change Notices, 2010), there are missing aspects of the metadata such as the measurement certainty and device used that are insufficient for every conceivable application of the data point. Furthermore, it is possible in an applied practice for an expert opinion to

determine the metadata of a data point for a particular use, regardless of the genesis of that data point. For both of these reasons, OptiRTC requires an operator to specify the metadata of every data point or time series in addition to the provenance-based metadata and a record of the calculations that describe it. Properties that can be specified include a unit of measurement, a range of expected values, a generic certainty parameter, and a text-based description.

Additionally, real-world constraints related to developer and operation costs make arguments for the user-imposed metadata model implemented in OptiRTC. Although OptiRTC is highly extensible, because of ongoing bandwidth and computing costs associated with querying web services, running models, and storing the inputs and outputs, new uses of the platform are dependent on financially-backed demand for the information service. Contracts for these services include time for an expert developer or engineer to verify the collection and processing of every measurement and calculation in the system. While it may be possible to construct an entirely provenance-based system of metadata management, the up-front developer costs associated with such an effort are incongruent with the available human resources for operating the system as dictated by the business model that funds its use.

The compromised result is a partial implementation of a provenance-based system, based around the fact that OptiRTC is administered via a software interface. A time series or data point that is mapped to a model parameter has its own provenance information, as do the decision space values that are produced when the model is run. The software administration tool is a simple database-editing program that provides integrated views of the components of an OptiRTC application, and an abstraction layer between the user and the nuances of database administration. Both features should improve the overall reliability of the system (#34). Additionally, it attempts to propagate input metadata forward to the metadata of the model output based on rules associated with each model in the sequence, but allows the user to modify any of these properties and stores this information separately from the input metadata. Furthermore, while metadata is recommended for every data point and series, OptiRTC does not require it for either inputs or decision space values. Above all else, the metadata management capabilities of OptiRTC are designed for compatibility with the workflow of its operators, which is an important component for the long-term acceptance of the platform by its operators (#38).

3.5 TIME SYNCHRONIZATION

A common issue with RTC and RTRM systems involves ensuring that all parts of the system are coordinated in time. Time coordination issues typically occur either when network delays cause previous information to arrive at its destination after more recent information from the same source, or when real-world conditions such as temperature affect the oscillation of the crystals of a controller's internal timer (Ingelrest et al., 2010). While this issue is exacerbated in RTC architectures that rely on multiple relay points between controllers or lack a central clock, it is a reality of many geographically distributed networks.

Because it uses a centralized RTC architecture, and time-stamping of data from in-the-field controllers is done via a web service with a similar centralized architecture, OptiRTC avoids many of the time synchronization issues other RTC systems encounter. Furthermore, although internet-based

communications inherently involve multiple relay points, because data transmission is typically two orders of magnitude faster than the 10-second maximum theoretical temporal resolution OptiRTC can facilitate, which itself is faster than many meaningful changes in the environmental systems it is designed to monitor and control, delays in the communications system should not be an issue in OptiRTC applications. Water does not flow that fast.

To avoid problems related to time zones and daylight savings time, all temporal data in OptiRTC is stored in Coordinated Universal Time (UTC). This convention also facilitates easier system maintenance, as the Network Time Protocol used by the Windows Time Service (W32Time) uses UTC time coordinates when synchronizing each piece of the OptiRTC hardware with an atomic clock. Although W32Time has a 1-2 second range of error, because the temporal resolution of the ioBridge-204 Pro data service is 10 seconds this range of inaccuracy is currently tolerable for OptiRTC applications. The W32Time service is provided by Microsoft and is a widespread convention of the internet, as it is the default time-synchronization method of most Microsoft operating systems.

3.6 OPTIRTC SECURITY

The security of OptiRTC is difficult to know, as there have not been any known breaches. All communications between OptiRTC and a Controller or System are incremental and HTTPS encrypted, and special queries exist to facilitate their quick retrieval. When such requests arrive at their destination, checks in the object serialization methods can possibly verify that records have not been intercepted and changed. Furthermore, all System access to any Controller configuration would require ioBridge authentication information that is not stored in the OptiRTC database. It is important to recognize that no networking system is completely secure. Where information is sent over areas that are not physically secured, it can be expected that individuals eventually start examining it. Although this information is encrypted, it would be wrong to assume that the information is completely protected from users with sufficient motivation and means. At the end of the day, what this means is that there are probably some RTC scenarios where a different

There are two security risks inherent in the OptiRTC platform as a result of the ioBridge controller. The first is related to the fact that the actual settings of the PLCs can be manipulated via a web interface if the correct credentials are provided. Even if these credentials never make it into the OptiRTC database, protecting them and their content is a top priority. Reinforcing this effort is the ability to compare ioBridge configuration details stored in the provenance model of the Controllers with the actual values of an ioBridge as reported by an authenticated web browser. By signing in and running a test, OptiRTC could detect if a change in either mechanism had occurred and alert an OptiRTC administrator. This functionality would require either a manual user login to avoid having to record the login credentials to the ioBridge service in OptiRTC resources. It is possible that this component of the system will be made to run independently, both in software and hardware, of OptiRTC, to mitigate the risks it might impose.

Generally, the same service oriented architecture that underlies many of OptiRTCs scalability potential also creates a larger external surface area for the disruption of the system. While these risks and tradeoffs are real, it should be noted that similar system architectures are crucial to the business model

of many large technology and e-commerce corporations. By depending on the same underlying technology as these much bigger players, OptiRTC is positioned to leverage the results of security investments made by much larger research teams and budgets than are available to dedicate to the development of OptiRTC.

Furthermore, the fact that ioBridge uses a cloud-based service can be seen as a security liability. Conventional wisdom in enterprise database administration stipulates that because cloud computing inherently shares hardware and bandwidth with an unknown number and type of other applications, it is inherently insecure. Although there have not been any ioBridge security incidents in any ioBridge applications, and other companies have created products around the PLC as well, this fear may be seen as problematic for certain applications. OptiRTC will clearly be applied in cases where the benefits accrued by adding real-time control outweigh these risks.

3.7 DDOE

Unfortunately, due to delays involving the non-RTC aspects of the project, installation of the DDOE RTC project has been delayed from this winter to the spring of 2011. At the time of this writing, not all of the known components that will be controlled have been confirmed. However, the system will employ two in-line cisterns with a sum of 5500 gallons of storage. This storage will be connected both to a line to a hose within the fire station for consumptive use, a line out to the city sewers for actuated releases, and a static line out to the city sewers to allow for passive overflow to occur.

The on-board logic for the system will be programmed to manage the actuation points of the system in response to conditions of the resources being managed, network connectivity conditions, power conditions, and set points provided to the Controller by OptiRTC. OptiRTC will use the National Weather Service's QPF and POP statistics, in concert with an understanding of the storage dimensions, to generate set-points for the storage, which will allow the cistern to be drawn down to different levels prior to a storm depending on the certainty and quantity of the forecasted precipitation. The initial method for determining set points will incorporate two independent parameters. The action threshold parameter will specify the forecast certainty required for OptiRTC to take an action. The conservation coefficient threshold will allow the system to be adjusted to exaggerate or limit the volume of water released in response to the expected volume of precipitation. The initial parameterization will set the action threshold at 70% and the conservation coefficient at 1.0. To keep the logic simple in the initial deployment, only the next hour's precipitation forecast will be used in the decision making logic. Every minute OptiRTC will look at the volume of water in the cisterns and calculate the void space. It will then compare this void space to the volume of rainfall that is expected to occur in the next hour, which will be determined by a computation involving the QPF, the known watershed area, and a runoff coefficient based on the drainage basin, which will be determined during the installation of the system. If the void space is less than the expected volume of precipitation, and the certainty of that precipitation is at least 70%, then OptiRTC will send a set point to the Controller that, in concert with the on-board logic, will prompt the Controller to actuate the CSO release valve to adjust the storage in the cisterns. The capacity created by this drawdown will reduce the peak discharge rates through the combined sewer during storm events, reducing the risk for sewer overflow.

4.0 CONCLUSION

OptiRTC is a highly flexible and extensible platform capable of reliably supporting a wide variety of environmental monitoring and control applications at a cost much less than comparable solutions. Its design incorporates a number of general features capable of being quickly configured to solve many problems in environmental engineering practice that can be adequately characterized by a 1-minute decision-making frequency, including combined sewer overflow problems in urbanized watersheds. Its architecture leverages internet technologies, arguably the most widely-adopted communications system ever created, to achieve the necessary scalability and performance needed to meet the demands of these kinds of projects.

OptiRTC represents an evolution of the concept of environmental engineering practice, and the sustainability of the infrastructure that results. Rather than conducting all design and engineering of a given resource management solution prior to installation, OptiRTC provides mechanisms for adapting the operating conditions of the infrastructure it controls at any point in the lifetime of that infrastructure. This is an extremely significant idea. It is reasonable to expect that the demands placed on water and other resource managers in a future of growing resource scarcity and regulatory complexity will require tools with adaptive capacity. At a higher level though, separating the physical form of infrastructure from the affect it has on the physical world by incorporating real-time decision making and control means that a wider range of conditions can be met with the same raw materials. Assuming that sourcing, designing, transporting, and building infrastructure is more resource-intensive than reprogramming it, and that raw materials are finite, the paradigm of introducing programmability into the design of building-scale environmental infrastructure has the potential to improve the sustainability of the environmental infrastructure by reducing the resource expenditure required for a retrofit. While controllable infrastructure itself is not a new paradigm, implementing it with standard enterprise networking and software components allows it to be implemented at a much lower cost, and maintained by individuals in a much larger workforce than the proprietary SCADA-based systems that are currently employed for environmental infrastructure control. Because of the scalability and flexibility of the computing architecture, the cost of the endpoint hardware, and ubiquity of the skills required to maintain its standards-based implementation, OptiRTC is a genuine attempt to create a distributed physical computing platform for environmental infrastructure that could be deployed at the scale necessary for the coordination of many small-scale environmental systems. Whether and in what conditions the global coordination this makes possible is able to supplant large-scale infrastructure projects is an open research question that, without this kind of platform, could not be adequately addressed.

APPENDIX A: NATIONAL WEATHER SERVICE DAILY TEMPERATURE PREDICTION REST SERVICE XML RESPONSE


```
<?xml version="1.0"?>
<dwml version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.nws.noaa.gov/forecasts/xml/DWMLgen/
schema/DWML.xsd">
  <head>
    <product srsName="WGS 1984" concise-name="time-series" operational-
mode="official">
      <title>NOAA's National Weather Service Forecast Data</title>
      <field>meteorological</field>
      <category>forecast</category>
      <creation-date refresh-frequency="PT1H">2010-12-01T19:30:47Z</creation-
date>
    </product>
    <source>
      <more-information>http://www.nws.noaa.gov/forecasts/xml/</more-
information>
      <production-center>Meteorological Development Laboratory<sub-
center>Product Generation Branch</sub-center></production-center>
      <disclaimer>http://www.nws.noaa.gov/disclaimer.html</disclaimer>
      <credit>http://www.weather.gov/</credit>
      <credit-logo>http://www.weather.gov/images/xml_logo.gif</credit-logo>
      <feedback>http://www.weather.gov/feedback.php</feedback>
    </source>
  </head>
  <data>
    <location>
      <location-key>point1</location-key>
      <point latitude="38.99" longitude="-77.01"/>
    </location>
    <moreWeatherInformation applicable-
location="point1">http://forecast.weather.gov/MapClick.php?textField1=38.99&a
mp;textField2=-77.01</moreWeatherInformation>
    <time-layout time-coordinate="local" summarization="none">
      <layout-key>k-p24h-n7-1</layout-key>
      <start-valid-time>2010-12-01T07:00:00-05:00</start-valid-time>
      <end-valid-time>2010-12-01T19:00:00-05:00</end-valid-time>
      <start-valid-time>2010-12-02T07:00:00-05:00</start-valid-time>
      <end-valid-time>2010-12-02T19:00:00-05:00</end-valid-time>
      <start-valid-time>2010-12-03T07:00:00-05:00</start-valid-time>
      <end-valid-time>2010-12-03T19:00:00-05:00</end-valid-time>
      <start-valid-time>2010-12-04T07:00:00-05:00</start-valid-time>
      <end-valid-time>2010-12-04T19:00:00-05:00</end-valid-time>
      <start-valid-time>2010-12-05T07:00:00-05:00</start-valid-time>
      <end-valid-time>2010-12-05T19:00:00-05:00</end-valid-time>
      <start-valid-time>2010-12-06T07:00:00-05:00</start-valid-time>
      <end-valid-time>2010-12-06T19:00:00-05:00</end-valid-time>
      <start-valid-time>2010-12-07T07:00:00-05:00</start-valid-time>
      <end-valid-time>2010-12-07T19:00:00-05:00</end-valid-time>
    </time-layout>
    <time-layout time-coordinate="local" summarization="none">
      <layout-key>k-p24h-n6-2</layout-key>
      <start-valid-time>2010-12-01T19:00:00-05:00</start-valid-time>
```

```
      <end-valid-time>2010-12-02T08:00:00-05:00</end-valid-time>
      <start-valid-time>2010-12-02T19:00:00-05:00</start-valid-time>
      <end-valid-time>2010-12-03T08:00:00-05:00</end-valid-time>
      <start-valid-time>2010-12-03T19:00:00-05:00</start-valid-time>
      <end-valid-time>2010-12-04T08:00:00-05:00</end-valid-time>
      <start-valid-time>2010-12-04T19:00:00-05:00</start-valid-time>
      <end-valid-time>2010-12-05T08:00:00-05:00</end-valid-time>
      <start-valid-time>2010-12-05T19:00:00-05:00</start-valid-time>
      <end-valid-time>2010-12-06T08:00:00-05:00</end-valid-time>
      <start-valid-time>2010-12-06T19:00:00-05:00</start-valid-time>
      <end-valid-time>2010-12-07T08:00:00-05:00</end-valid-time>
    </time-layout>
    <parameters applicable-location="point1">
      <temperature type="maximum" units="Fahrenheit" time-layout="k-p24h-n7-
1">
        <name>Daily Maximum Temperature</name>
        <value>57</value>
        <value>46</value>
        <value>44</value>
        <value>42</value>
        <value>42</value>
        <value>42</value>
        <value>42</value>
      </temperature>
      <temperature type="minimum" units="Fahrenheit" time-layout="k-p24h-n6-
2">
        <name>Daily Minimum Temperature</name>
        <value>29</value>
        <value>26</value>
        <value>26</value>
        <value>25</value>
        <value>25</value>
        <value>24</value>
      </temperature>
    </parameters>
  </data>
</dwml>
```


Accessed on 12/1/2010 at 2:40 PM Eastern Standard Time via:
http://www.weather.gov/forecasts/xml/sample_products/browser_interface/ndfdXMLclient.php?lat=38.99&lon=-77.01&product=time-series&begin=2004-01-01T00:00:00&end=2013-04-20T00:00:00&maxt=maxt&mint=mint

REFERENCES

Boi Faltings, *Learning to Cope with an Open World*. Artificial Intelligence Laboratory, Swiss Federal Institute of Technology, IN-Ecublens, 1015 Lausanne, Switzerland, (1995). Accessed via <<citeseerx.ist.psu.edu>

Deshpande, A., Nath, S., Gibbons, P.B., Seshan, S., *Cache-and-Query for Wide Area Sensor Databases.* Proc. 22nd ACM SIGMOD Int. Conf. Management of Data Principles of Database, (2003).

Duchesne, S., Mailhot, A., Dequidt, E., Villeneuve, J.P., *Mathematical modeling of sewers under surcharge for real time control of combined sewer overflows,* Urban Water, Vol. 3, pp. 241-252, (2001).

Glasgow, H., Burkholder, J.M., Reed, R., Lewitus, A., Kleinman, J., *Real-time remote monitoring of water quality: a review of current applications, and advancements in sensor, telemetry, and computing technologies.* Journal of Experimental Marine Biology and Ecology Vol. 300, pp. 409-448, (2004).

Ingelrest, F., Barrenetxea, G., Schaefer, G., Vetterli, M., Couach, O., *SensorScope: Application-Specific Sensor Network for Environmental Monitoring. ACM Transactions on Sensor Networks*, Vol. 6, No. 2, Article 17, Publication date: February 2010, ACM Transactions on Sensor Networks (2010).

Lehning, M., Dawes, Bavay, M., WSL Institute for Snow and Avalanche Research SLF; Parlange, M, Ecole Polytechnique Federale de Lausanne; Nath., S, Zhao, F, Microsoft Research, *Instrumenting the Earth: Next-Generation Sensor Networks and Environmental Science*, Part 1, Essay 6, pp. 45-54, *The Fourth Paradigm: Data-Intensive Scientific Discovery.* Microsoft Research, 2009. Accessed via <<http://research.microsoft.com/en-us/collaboration/fourthparadigm/contents.aspx>>

Meirlaen, J., Benedetti, L., *Modelling and real-time control of the integrated urban wastewater system.* Environmental Modelling & Software 20, pp. 427-442, (2005).

Nyberg, U., Andersson, B., Aspegren, H., *Real time control for minimizing effluent concentrations during storm water events*. Water, Science and Technology, Vol. 34, No. 3-4, pp. 127-134, (1996).

Pleau, M., Golas, H., Lavallee, P., Pelletier, P., Bonin, R., *Global optimal real-time control of the Quebec urban drainage system*. Environmental Modelling & Software, Vol. 20, pp. 401-413, (2005).

Quinn, N.W.T., Hanna, W.M., *A decision support system for adaptive real-time management of seasonal wetlands in California.* Environmental Modelling & Software, Vol. 18, pp. 503-511, (2003).

Rossman, L., *Storm Water Management Model User's Manual Version 5.0.* Water Supply and Water Resources Division, National Risk Management Research Laboratory, United States Environmental Protection Agency, Cincinnati, OH. Revised July 2010. Accessed via <http://www.epa.gov/ednnrmrl/models/swmm/epaswmm5_user_manual.pdf>

Ruggaber, T., Talley, J., Montestruque, L.A., *Using Embedded Sensor Networks to Monitor, Control, and Reduce CSO Events: A Pilot Study.* Environmental Engineering Science, Vol. 24, Number 2, (2007).

Schutze, M., Alex, J., *Advanced pumping station control for RTC of sewer systems.* Pumps, Electromechanical Devices and Systems Applied to Urban Water Management, Vol. 2, pp. 887–894, (2003).

Schutze, M. Campisano, A., Colas, H., Schilling, W., Vanrolleghem, P., *Real time control of urban wastewater systems - where do we stand today?* Journal of Hydrology, Vol. 299, pp. 335-348, (2004).

Vaughan IP; Diamond M; Gurnell AM., *Integrating ecology with hydromorphology: a priority for river science and managment*. Aquatic Conservation-Marine and Freshwater Ecosystems, Vol. 19, No. 1, pp. 113-125, (2009).

Venkataraman, S., Benger, W., Long, A., Jeong, B., Renambot, L., *Visualizing Hurricane Katrina - Large Data Management, Rendering, and Display Challenges.* Center for Computation and Technology, Louisiana State University; Association for Computing Machinery (2006).

Maximum Capacity Specifications for SQL Server 2005, Microsoft Developer Network, Microsoft Corporation. Last Updated 9/15/2007. Accessed via <<http://msdn.microsoft.com/en-us/library/ms143432(v=SQL.90).aspx>>

National Weather Services XML Services Change Notices, National Oceanic and Atmospheric Administration, 2010: http://www.weather.gov/forecasts/xml/xml_changes.xml

Support boundary to configure the Windows Time service for high accuracy environments, Microsoft Support, Microsoft Corporation. Article ID: 939322, Revision 3.0, Last Reviewed 2/2/2010. Accessed via <<http://support.microsoft.com/kb/939322>>

Windows Time Service, How It Works, Windows Server TechCenter, Microsoft TechNet, Microsoft Corporation. Last Updated 3/12/2010. Accessed via <<http://technet.microsoft.com/en-us/library/cc773013(WS.10).aspx>>