Safety Verification of Floodgate Operation Protocols Using Hybrid Automata

M. V. Panduranga Rao, Akhilesh Chaganti

***Abstract*—Flooding is one of the most damaging of natural disasters. Structural approaches to flood management consist of reservoirs and dams equipped with floodgates, along with protocols for their operation. However, in spite of the infrastructure being in place, floods can occur because of flaws in the floodgate operation protocols or human error in its implementation.**

Hybrid automata are a formalism for modeling systems that have discrete as well as continuous components. In some cases it is possible to efficiently decide whether such systems satisfy precisely defined safety criteria. We model a class of flood management systems as hybrid automata and use existing verification techniques to prove safety of floodgate operation protocols. This approach yields a tool for evaluating such protocols. To the best of our knowledge, this is the first application of formal methods in computer science to the problem of flood management.

***Index Terms*—Flood Management, Hybrid Systems, Formal Methods, Hybrid Automata, Formal Verification of Hybrid Systems**

I. Introduction

Every year, thousands of lives and property worth billions of dollars are lost in floods all over the world [1]. Therefore, a lot of resources and effort are spent in flood prevention, mitigation and recovery.

Structural approaches to flood management systems typically involve a system of reservoirs and dams with floodgates to control the flow between the reservoirs [13].

In the case of multipurpose dams and reservoirs, there are conflicting objectives in play. On one hand reservoirs need to store water for purposes like irrigation and water supply. On the other hand, reservoirs are also used as a measure against flooding of locations downstream on the river. If too much water is stored, the ability of the dam to shield the downstream locations diminishes.

Generally, floodgate managers have carefully designed manuals and protocols that guide them through the procedure for operating the floodgates during different situations. In spite of having these flood management systems in place, there have been several incidents where an error in operation of the floodgates have resulted in flooding. Errors are chiefly in misjudging the timing of actuation of the floodgates. The main problem is that a lapse during the protocol design stage can result in a disaster–either due to an inherent problem with the protocol or due to a potentially dangerous sequence of events that was not envisaged during the protocol design phase. Moreover, in case such a sequence of events does occur, the operators do not have a backup plan. Indeed, such a protocol was suspected to be a reason for the Wivenhoe and Somerset dam disaster in Australia in 2010 [3].

For a motivating example where timing of floodgate operation makes a difference, consider a sequence of three reservoirs connected by water channels. There are floodgates at each reservoir that open into the channel that takes the water downstream. There are upper thresholds for each of these reservoirs beyond which they flood. Associated with each channel is a delay required by the water to travel across that channel and associated with each floodgate is an opening delay.

In this infrastructural setting, following are some sequences of events that can lead to the flooding of at least one of the second and third reservoirs:

- Suppose that the water levels at the second and the third reservoir are very close to their respective upper thresholds and are rising. Suppose the water level at the second reservoir demands that its floodgate should be opened. This decision also makes sense in the knowledge that the water level at the third reservoir is below the threshold at least for the moment. After elapse of the channel delay, the water released from the second reservoir reaches the third reservoir. But by this time, the level at the third reservoir might have reached the threshold, and the third reservoir floods.
- In order to avoid the above scenario, the second reservoir can choose to delay the release of its water to the third reservoir as much as possible. But then, a release of water from the first reservoir can flood the second one. This problem is exacerbated by the fact that there is a delay associated with the opening of the floodgate connecting the second reservoir to the third reservoir.

Thus, the decision of when to open the floodgates is a function of infrastructural parameters of the reservoir-floodgate system like the channel delays and flooding thresholds and dynamical inputs from sensors like the current water level at each reservoir and the rate of its change.

Since reservoirs differ in these parameters, a uniform policy like "open all the floodgates whenever there is a rise in water levels at any reservoir" may not work in all scenarios. Indeed, the fact that there can be a drop in the water levels of a reservoir when its floodgate is open offers more flexibility in decision making.

The objective of this work is to be able to establish whether a given floodgate operating strategy is safe. That is, whether it always keeps the water levels within threshold limits at all reservoirs.

It is often important to be able to formally verify if a software or a hardware system behaves as expected, particularly in safety critical systems. Since modern systems are large and complicated, manual verification can be extremely tedious

Manuscript received January 08, 2014; revised February 06, 2014.

This work was supported by the Indo-Japanese project on "Information Network for Natural Disaster Mitigation and Recovery" (SATREPS-DISANET) and the DIT, Govt. of India sponsored project on "Cyber Physical Systems" at IIT Hyderabad.

M.V. Panduranga Rao is with the Dept. of CSE, IIT Hyderabad, India. Email: mvp@iith.ac.in. Akhilesh Chaganti is with the Dept. of CS, Stony Brook University. Email: akhilesh.chaganti@stonybrook.edu.

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

and error prone. It is therefore natural to look for algorithmic techniques that automatically do this verification for us. This requires us to work with mathematical formalisms to model both the system and the behaviour required of the system in a precise and unambiguous manner. Algorithms can then be sought that verify whether the given (model of the) system satisfies the specified requirements.

A popular formalism for describing systems that have both discrete and continuous components is the hybrid automaton [7]. Efficient algorithms exist for verifying restricted classes of hybrid automata, yielding verification tools like HyTech [9]. In this paper we propose modeling of reservoir-floodgate systems as hybrid automata and apply HyTech to verify whether or not safety criteria are met.

The rest of the paper is organized as follows. The next section formally defines the problem that we address in this paper. Section III gives a brief introduction to hybrid systems, hybrid automata and the verification problem for hybrid automata. The solution of the problem through hybrid automata is discussed in section IV. Section V concludes the paper and discusses open problems.

II. Problem Statement

Suppose there is a series of n reservoirs $v_1, \ldots, v_n$. The last reservoir v_n drains into the river, labeled v_{n+1}. There is a water channel $e_{i,i+1}$ between v_i and v_{i+1} for $1 \leq i \leq n$. Water always flows from v_i to v_{i+1} and not in the other direction.[1]

There is a floodgate g_i at reservoir v_i installed at the beginning of the channel $e_{i,i+1}$. The floodgate g_i can be opened, which results in a flow of water from v_i to v_{i+1} at a rate $f_{i,i+1}$ units of volume per unit time (for $1 \leq i \leq n$). When it is closed, the flow stops.

For the sake of simplicity, in this paper we work with the assumption that the outflow rate from a reservoir can remain the same regardless of whether there is an inflow from the upstream reservoir or not. Indeed, this is possible for floodgates with adjustable outflow rates.

Each reservoir v_i has an upper threshold of water level U_i associated with it beyond which if the water level rises, the reservoir floods. However, a floodgate operating protocol can assume a more conservative upper threshold $u_i \leq U_i$, as will be seen later. There is also a lower threshold (say, 0).

For each channel $e_{i,i+1}$, there exists a delay of $d_{i,i+1}$ time units for the water to travel from v_i to v_{i+1}. Thus, when water is released from v_i, it reaches v_{i+1} after $d_{i,i+1}$ time units. Finally, associated with each floodgate g_i is a delay of t_i time units incurred for opening the floodgate. Note that these parameters are infrastructural in nature.

We denote by x_i and $\dot{x}_i$ the current water level and its rate of change respectively at reservoir v_i. Thus, if precipitation at v_i is p_i,

$$\dot{x}_i = p_i + f_{i-1,i} - f_{i,i+1}$$

for $2 \leq i \leq n$ and $\dot{x}_i = p_i - f_{i,i+1}$ for $i = 1$. These dynamical quantities are typically collected by sensors. We assume that each reservoir is equipped with sensors that report these parameters. All the sensor data is transmitted to a central control room. The control room can actuate the floodgates into opening or closing.

A *strategy* is a set of decisions regarding when (in terms of water levels) to open and close floodgates at different reservoirs keeping in view the infrastructural and dynamical parameters. A *safe strategy* is one that never results in water level exceeding the safety limit at any reservoir. The problem that we address in this paper is to decide whether a given strategy is safe.

[1]While such a topology is restrictive, several flood management systems are actually linear. A famous example is the Tokyo Metropolitan Area Outer Underground Discharge Channel also called the G-Cans project [2].

III. Hybrid Automata: Modeling and Verification

In this section, we give a brief and informal description of hybrid automata and its use in verification of hybrid systems. We will illustrate the salient features of hybrid automata using an example. Please see [7] for a formal definition. We assume familiarity with elementary definitions and terminology in graph theory [6] and formal logic [11].

A hybrid automaton H is defined by the following entities:

- A finite directed multigraph $G = (V, E)$. The vertices of the multigraph model the different *modes* in which the discrete component of the system functions.
- A finite ordered set $X = \{x_1, x_2, \ldots, x_n\}$ of real valued variables for modeling the continuous components.
- An initial mode and an assignment of initial values to variables in X. The automaton starts in this mode, with real variables having these values.
- For each mode $v \in V$, a specification of flow conditions for variables in X. This is typically done using differential equations that govern the evolution of the real valued variables when the system is in the mode v.
- For each mode $v \in V$, a specification of invariance requirements on variables in X. These requirements are specified as finite conjunctions of linear inequalities on the real valued variables.
- Associated with each directed edge $e = (v, w) \in E$ is a predicate $jump(e)$ that specifies the range of values that variables in X can have when jumping from v and the values that the variables can assume on completing the jump to w.

Complex hybrid systems that consist of several communicating hybrid systems executing concurrently can be modeled by composing hybrid automata and using appropriate mechanisms for synchronization and message passing. In this paper, we use the shared variables feature of hybrid automata for this purpose.

Fig. 1 depicts a hybrid automaton. This automaton is constructed in the context of the reservoir-floodgate safety verification problem; the semantics of the construction will be clear shortly when the solution is discussed. For now, we use it to explain some important features of hybrid automata. The automaton multigraph consists of three modes $Fill1$, $Open1Delay$ and $Drain1$ that model the discrete component. The continuous component is modeled by the real valued variables x_1, t_1, and k_1. The initial mode is $Fill1$, with the real valued variables being assigned initial values as $\{x_1 := 35, t_1 := 0, k_1 := 0\}$. Thus, the system starts in $Fill1$ with the real valued variables x_1, t_1 and k_1 having values 35, 0 and 0 respectively. Further, for $Fill1$,

- The flow conditions are defined by $\{\dot{x}_1 = 5, \dot{t}_1 = 0\}$ – when in mode $Fill1$, the rate of change in the real

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

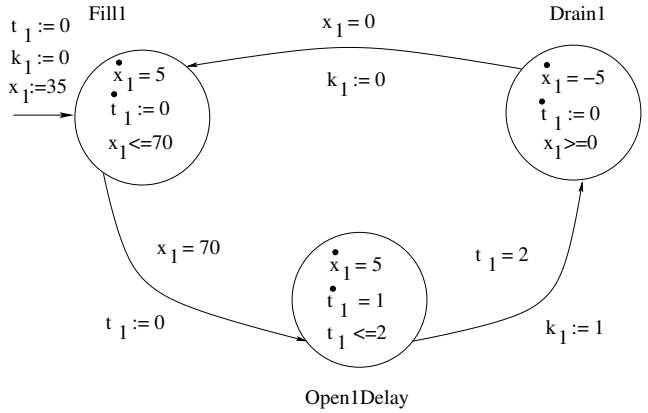

Fig. 1. Hybrid automaton $reservoir1$ for the first reservoir

valued variable x_1 is 5 units per unit time and the rate of change of t_1 is 0.[2]

- The invariance condition is defined by $\{x_1 \leq 70\}$ – when in mode $Fill1$, x_1 must be at most 70.
- $jump(e) = \{(x_1 = 70, x_1 := 70, t_1 = 0, t_1 := 0, k_1 = 0, k_1 := 0\}$ – when in mode v, it is legal to jump across the directed edge e to the destination mode ($Open1Delay$) when $x_1 = 70$. On completing the jump, x_1 will assume the value 70 and t_1 and k_1 will retain their values.

Behaviour expected of the system is specified using some variant of temporal logic [11]. In this paper, we restrict ourselves to conjunctions of linear equalities like $x_1 \leq 100$ & $x_2 \leq 100$ which will mean that we require the real valued variables x_1 and x_2 to never exceed 100.

The verification problem is to decide if the system thus modeled satisfies the formally specified behaviour expected of the system.

Over the past two decades, a lot of research has been done in the field of verification of hybrid systems [4], [7], [9]. It has been shown that efficient verification algorithms exist for a class of hybrid automata called linear hybrid automata [7]. Indeed, verification tools based on these ideas exist and are being developed over the past two decades. In this paper, we use such a tool by Henzinger et al. called HyTech [9], [8]. These tools have been extensively used in the past for scientific and industrial applications [5], [10], [12], [14].

IV. The Hybrid Systems Approach

We model each reservoir-floodgate system as a separate hybrid automaton. The overall system is obtained by composing the hybrid automata for the individual reservoir-floodgate systems. Synchronization between different reservoir-floodgate systems is achieved through the shared variables feature of HyTech.

A. Details

We illustrate our approach using the three reservoir system discussed previously. All the reservoirs have a flooding threshold of $U_i = 100$ units. Suppose that the strategy adopted is that a floodgate g_i should be opened when the water level reaches $u_i = 70$ units. We wish to verify if this strategy keeps all the reservoirs safe or not for different values of the other parameters.

The first reservoir, which is most upstream, is modeled by the automaton *reservoir1* (Fig. 1). In the example shown, the initial water level is 35 units, and there are three discrete modes of operation. The system is initially in the mode $Fill1$ and the water rises at the rate of 5 units per unit time. The real valued variable x_1 stands for the instantaneous water level at the reservoir. As mentioned previously, the strategic operational upper threshold u_1 for water level for this reservoir is 70 units. Hence the invariant condition for $Fill1$ says that x_1 should be less than or equal to 70 units.

When the level reaches 70 units, the floodgate is opened. As discussed in section II, there exists a delay in opening the floodgate. The automaton jumps to the mode $Open1Delay$.

[2]Although k_1 is technically a real valued variable, the verification tool that we use in this paper allows declaration of discrete variables. Discrete variables have flow rates of 0, and need not be specified explicitly. We use this discrete variable merely as a shared variable to realize synchronization between different hybrid automata.

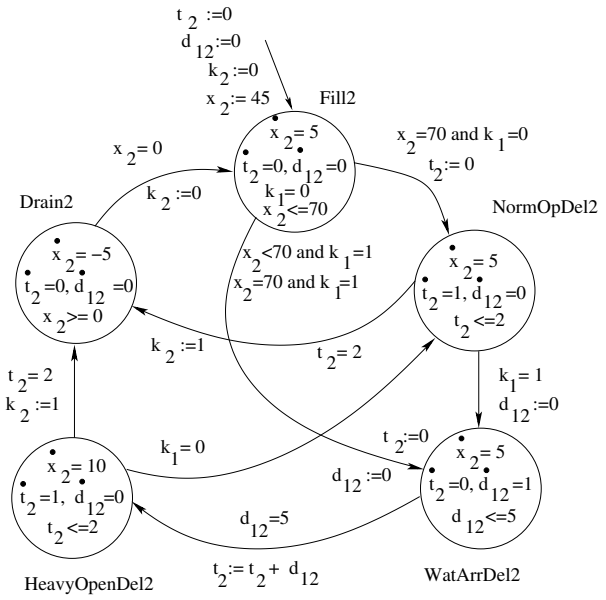

Fig. 2. Hybrid automaton $reservoir2$ for the second reservoir

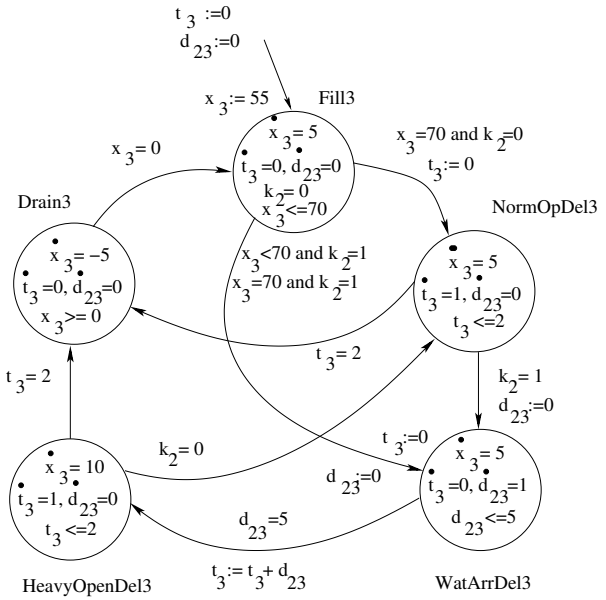

Fig. 3. Hybrid automaton $reservoir3$ for the third reservoir

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

The clock t_1 counts out the delay (of 2 time units in the example), during which time the water continues to rise at the previous rate. The floodgate opens at the elapse of the delay, and water drains out into the channel. Simultaneously, a shared variable k_1 is set to 1 to inform the downstream reservoir automaton that water has been released. Indeed, the automata for different reservoir-floodgate systems synchronize on these shared variables. When the floodgate is open, water flows out to the downstream reservoir (at the rate of -5 units per unit time in this example). When the level reaches 0, the floodgate closes. This fact is conveyed to the downstream automaton by setting k_1 to 0, and the automaton switches mode to $Fill1$ again.

The automaton for the second reservoir, $reservoir2$, is somewhat different because of the fact that it is downstream to $reservoir1$ but upstream to $reservoir3$. In the example shown (Fig. 2), this reservoir is also in the mode $Fill2$ initially, with the initial water level at 45 units, and rising at 5 units per unit time.

This normal rate of filling up of the reservoir can be disturbed by two events–the water reaching the upper threshold or the water released by the upstream reservoir reaching $reservoir2$. The rate of rise of the water levels differs for these two scenarios. In the first case, it remains the same. In the second case, the water coming in from $reservoir1$ adds to the rise due to precipitation at the second reservoir, making it a total of 10 units per time unit. We need to distinguish between these two.

As shown in the example, if the water reaches the upper threshold u_2 of its own accord, the automaton jumps to $NormOpDel2$, where it waits for (2 time units in this example) the floodgate to open, and then transits to $Drain2$, wherein the water is drained into the third reservoir. Note that the rate of rise of the water level while the automaton is waiting for the floodgate to open continues to be the same as it is in the $Fill2$ mode.

If $reservoir2$ gets to know through the shared variable k_1 that the floodgate at $reservoir1$ has been opened, it jumps to $WatArrDel2$ where it waits for the water to reach from the first reservoir to the second. In this mode, the rate of rise of water continues to remain 5 units per unit time. Note that this jump can happen from both the modes $Fill1$ or $NormOpDel2$.

After the delay elapses, the system transits to the mode $HeavyOpenDel2$, where the rate of rise of water includes the water flow from the upstream reservoir and is therefore 10 units per unit time. The systems waits in this mode for the floodgate opening to complete. Meanwhile, if k_1 is set to 0 by $reservoir1$, the system reverts to the $NormOpDel2$ mode.

When the floodgate is fully open, it transits to the draining mode $Drain2$. Now the shared variable k_2 is set to 1 to notify the downstream reservoir $reservoir3$ that water has been released.

The automaton $reservoir3$ for the third reservoir in the example is similar (Fig. 3). The only difference is that since it has no reservoirs downstream to it and drains into the river, it need not set any shared variables.

Obviously, the safety criterion is specified by the predicate $x_1 \leq 100 \quad \& \quad x_2 \leq 100 \quad \& \quad x_3 \leq 100$.

Tables I and II illustrate some example scenarios and the corresponding safety verdicts. The scenario just discussed corresponds to the parameters labeled by P_2 and the strategy labeled by S_3 for the initial water levels $35, 45$ and 55 at the first, second and third reservoirs respectively. It can be seen from the table that the system is safe in this case. However, for the same set of parameters, the strategy is unsafe when the initial levels are 55, 45 and 35 units.

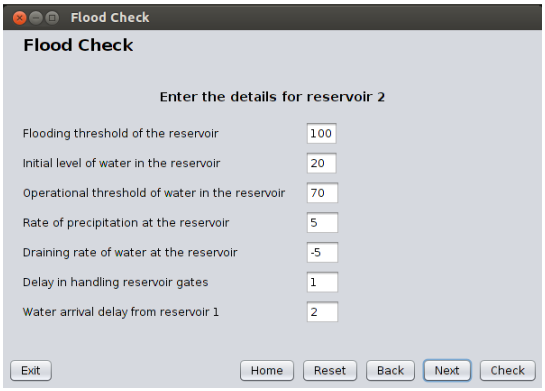

Fig. 4. Example screenshot of the interface for $reservoir2$

B. The Tool

It is now possible to write a simple java based tool with HyTech at the backend that fits into a larger floodgate management system. A GUI enables the operator at the central control room to enter for all reservoirs (i) infrastructural parameters like the flooding threshold of the reservoir, water outflow rate when a floodgate g_i is open, the delay $d_{i,i+1}$ for the channel $e_{i,i+1}$ and the gate opening delay t_i (ii) sensor inputs that report initial water levels x_i and rate of precipitation p_i and (iii) the strategic operational threshold u_i. Note that the water outflow rate at an upstream reservoir-floodgate system is an infrastructural input parameter for the downstream system.

The tool then generates a HyTech (.hy) file that contains the hybrid automata description and safety specifications, runs HyTech on the file and reports whether the strategy is safe or not. If the tool declares a strategy as safe, floodgates can be operated according to this strategy. Fig. 4 shows an example interface for $reservoir2$.

The block diagram in Fig. 5 illustrates a potential architecture of the entire system, with the floodgate management system in context.

V. Conclusions and Future Directions

In this paper, we applied ideas from formal methods in computer science to the problem of verification of floodgate management strategy. We demonstrated its effectiveness in a linear arrangement of reservoirs. Several directions of future research remain open. Many of these are essentially relaxations in the assumptions and generalizations of the problem statement that we addressed in this paper, yielding hybrid automata that are more faithful models to different systems:

- An interesting question is how this framework generalizes when the reservoirs are not in a linear arrangement. In other words, when the set of reservoirs are the nodes

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I

EXAMPLE INFRASTRUCTURAL AND DYNAMICAL PARAMETERS. COLUMNS LABELED BY T_1, T_2 AND T_3 INDICATE THE TIME DELAY INCURRED IN OPENING FLOODGATES g_1, g_2 AND g_3 RESPECTIVELY. SIMILARLY, D_{12} AND D_{23} INDICATE THE CHANNEL DELAY BETWEEN RESERVOIRS 1 AND 2 AND 2 AND 3 RESPECTIVELY.

	Infrastructural Parameters					**Sensor Inputs:** p_i, $p_i + f_{i-1,i}$, $p_i + f_{i-1,i} - f_{i,i+1}$		
Parameter	T_1	T_2	T_3	D_{12}	D_{23}	Reservoir 1	Reservoir 2	Reservoir 3
P_1	2	2	2	5	10	5,5,-5	5,10,-5	5,10,-5
P_2	2	2	2	5	5	5,5,-5	5,10,-5	5,10,-5
P_3	5	5	5	5	5	5,5,-5	5,10,-5	5,10,-5
P_4	3	2	2	5	5	5,5,-5	5,10,-5	5,10,-5
P_5	2	2	2	5	5	5,5,-5	10,15,-10	5,15,-5
P_6	2	2	2	5	5	10,10,-10	5,15,-5	5,10,-5
P_7	2	2	2	5	5	15,15,-15	5,20,-10	5,15,-5

TABLE II

SAFETY VERDICTS FOR THE EXAMPLE SCENARIOS WITH TWO SETS OF INITIAL VALUES. INITIAL LEVELS AT RESERVOIRS 1, 2 AND 3 ARE 55,45 AND 35 UNITS RESPECTIVELY IN THE FIRST SET AND 35,45 AND 55 FOR THE SECOND. THE STRATEGIES ARE DEFINED THUS: $S_1 : u_1 = 60, u_2 = 60, u_3 = 60$; $S_2 : u_1 = 60, u_2 = 70, u_3 = 80$; $S_3 : u_1 = 70, u_2 = 70, u_3 = 70$; $S_4 : u_1 = 80, u_2 = 70, u_3 = 60$; $S_5 : u_1 = 80, u_2 = 80, u_3 = 80$. WHETHER OR NOT A GIVEN STRATEGY IS SAFE UNDER DIFFERENT INFRASTRUCTURAL AND DYNAMICAL PARAMETERS IS GIVEN BY Y/N.

	Initial Levels: 35,45,55					**Initial Levels: 55,45,35**				
	S_1	S_2	S_3	S_4	S_5	S_1	S_2	S_3	S_4	S_5
P_1	Y	N	Y	Y	Y	N	Y	N	N	N
P_2	Y	N	Y	Y	Y	Y	Y	N	N	N
P_3	Y	N	Y	N	N	N	N	N	N	N
P_4	Y	N	Y	Y	Y	Y	Y	N	Y	N
P_5	Y	N	N	Y	N	N	N	Y	Y	N
P_6	Y	Y	N	N	Y	Y	Y	Y	Y	Y
P_7	Y	Y	Y	N	N	Y	Y	Y	N	N

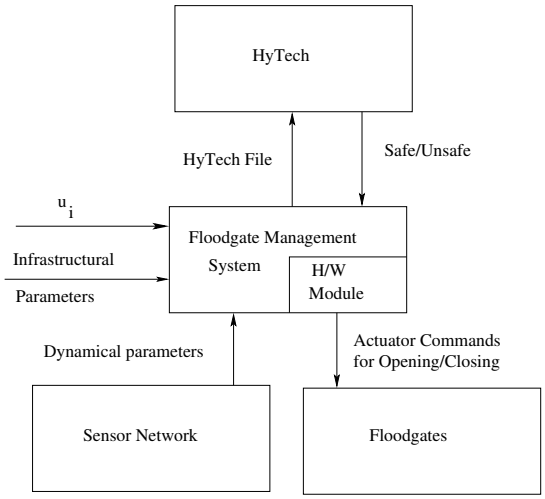

Fig. 5. Architecture of the Reservoir-Floodgate system.

of a general directed graph and the water channels are the edges.

- We have assumed that the rates of change of real valued variables (like water levels and flow from a floodgate) do not vary. How does one tackle the situation when they do vary?

REFERENCES

[1] Integrated urban flood management. http://www.cap-net.org/content/integrated-urban-flood-management/, 2004. Accessed: 2014-01-30.

[2] G-cans project. http://www.water-technology.net/projects/g-cans-project-tokyo-japan/, 2012. Accessed: 2014-01-30.

[3] Queensland floods commission of enquiry. http://www.floodcommission.qld.gov.au/publications/final-report, 2012. Accessed: 2014-01-30.

[4] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.*, 138(1):3–34, Feb. 1995.

[5] D. Bosscher, I. Polak, and F. W. Vaandrager. Verification of an audio control protocol. In *FTRTFT*, pages 170–192, 1994.

[6] R. Diestel. *Graph Theory (Graduate Texts in Mathematics)*. Springer, August 2005.

[7] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, LICS '96, pages 278–, Washington, DC, USA, 1996. IEEE Computer Society.

[8] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to hytech. In *TACAS*, pages 41–71, 1995.

[9] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: A model checker for hybrid systems. *International Journal on Software Tools for Technology Transfer*, 1(1-2):110–122, 1997.

[10] P.-H. Ho and H. Wong-Toi. Automated analysis of an audio control protocol. In *CAV*, pages 381–394, 1995.

[11] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, New York, NY, USA, 2004.

[12] A. Puri and P. Varaiya. Verification of hybrid systems using abstractions. In *Hybrid Systems*, pages 359–369, 1994.

[13] B. Rampano. Water control structures: Design suitability for natural resource management on coastal floodplains. 2009.

[14] T. Stauner, O. Müller, and M. Fuchs. Using hytech to verify an automative control system. In *Hybrid and Real-Time Systems, LNCS*, pages 139–153, 1997.

ISBN: 978-988-19252-5-1
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)