

ADA's Virtual Node based Water System Simulator

A.Hosseinzaman and A.Bargiela

Department of Computing
Nottingham Trent University
Burton Street, Nottingham NG1 4BU,U.K.

Abstract. This paper presents the application of Virtual Nodes for the distributed implementation of the nonlinear network tearing algorithm. The program is implemented on a network of SUN/SPARC workstations connected by an Ethernet communication link, having a UNIX operating system.

1. Introduction

This paper presents the main issues concerning the design and implementation experience of the nonlinear network tearing algorithm[2,3] programmed in Ada for distributed computing system using the Virtual Node concept.

The early telemetry systems employed in the water industry acquired and stored data from the remote-stations for the engineer's inspection. The data, which depicts a snap shot of the state of the water system, was used to assist the engineer in deciding operational controls of the water distribution system. The term used in water industry to describe such telemetry/telecontrol systems is Supervisory Control and Data Acquisition(SCADA).

However, the gradual growth of the water system resulted in the accumulation of the increasing volume of data which had to be attended to by the engineer. To help reduce the work load and achieve a more efficient and cost effective operational control of the system, simulation software were introduced[6,7]. The role of the simulation software was to analyze the system parameters and arrive at a decision which would then be accepted or modified by the engineer who is responsible for the system operations.

The integration of the simulation software with the existing telemetry systems into a real-time decision support system proved to be a challenging task due to the rapid increase of computational requirement of the simulation algorithms with increase of the physical network size. A classical solution to this problem, frequently adopted in the case of linear systems, was to partition the network into a number of subnetworks, solve the subnetworks in isolation and then coordinate the subsystem solutions in order to find the overall system solution[1]. This approach was developed and generalized for nonlinear networks in an earlier research[2,3].

Having developed the distributed diakoptics based simulation algorithm, we have focused on its efficient and flexible implementation on the distributed computing hardware. As the simulation software is envisaged to undergo continuous development in line with the

development of the physical telemetry systems, it has been important to select the implementation language that is particularly well suited to such a task.

Ada is a language that was specifically designed to be used for the development of large scale software systems. It provides strong typing, enforces public/private structure and provides explicit specification of parallelism. However, since Ada was not originally designed for distributed systems, it has been necessary to extend Ada's rendezvous mechanism to facilitate network communications. This mechanism is the Ada Virtual Node model described in [4].

2. Water System Simulation

The telemetry systems employed in water distribution networks consist of a number of remote-stations relaying information about flows and pressures at some discrete points in the network. However, to get a global view of the water distribution system this data needs to be interpreted in conjunction with the mathematical model of the network. The model is constructed by applying the basic rules of continuity to water networks. Mathematically the model represents a set of simultaneous nonlinear equations depicting the inflows and outflows at each node or flow between two nodes of the water network. The form of the relationship depends on the physical characteristics of the element under consideration (e.g. pipe, valve, pumps etc). The system governing equations can be formulated in accordance with the following rules: (i) nodal mass balance - the algebraic sum of all the inflow and outflow at each node is equal to zero, and (ii) energy conservation - the total sum of all the head losses around any loop in the network is equal to zero.

Using these rules and the functional relationships between flow and pressure drop for every link of the network, we arrive at the pipe flow f_{ij} equation which depends nonlinearly on the pressure difference at the end-nodes of the pipe,

$$f_{ij}(x) = R_{ij}(x_i - x_j)^{0.54} \quad (1)$$

where R_{ij} is the hydraulic resistance of the i-j pipe and the vector of the nodal pressures is $x = \{ x_1 \dots x_n \}^T$; n is the number of nodes in the network.

The nodal pressures x are usually calculated from mass balance equations in n-1 network nodes and one reference pressure in an arbitrarily selected n-th node.

$$g_i(x) = \sum_{j \in \Omega_i} f_{ij}(x) \quad (2)$$

$$\mathbf{g}_i(x) = x_n \quad (3)$$

where $\mathbf{g}_i()$ is a mass balance in node i , Ω_i is a set of nodes adjacent to node i .

In the network without storage elements, $\mathbf{g}_i()$ corresponds to the consumption/supply out node i , for $i=\{1, \dots, n-1\}$, and to a measurement of the reference pressure in node n for $i=n$,

$$Z_i = \mathbf{g}_i(x) \quad (4)$$

The system of nonlinear equations to be solved can therefore be represented in a compact form as:

$$Z = \mathbf{g}(x) \quad (5)$$

where $Z = [Z_1 \dots Z_{n-1}, Z_n]^T$ and $\mathbf{g}(x) = [\mathbf{g}_1(x) \dots \mathbf{g}_n(x)]^T$.

The solution of (5) involves linearization of the system of equations and iterative improvement of the initial estimate of the vector x , x_0 .

$$\mathbf{g}(x) = \mathbf{g}(x_0) + \left. \frac{\delta \mathbf{g}}{\delta x} \right|_{x_0} \Delta x \quad (6)$$

introducing $J = (\delta \mathbf{g} / \delta x)$ for the Jacobian matrix and noting (5),

$$Z = \mathbf{g}(x_0) + J \Delta x \quad (7)$$

So

$$\Delta x = J^{-1}(Z - \mathbf{g}(x_0)) \quad (8)$$

and the iterative solution is obtained as:

$$x^{k+1} = x^k + \Delta x \quad (9)$$

The nonlinearity of the equations calls for an iterative method of solution such as

Newton_Raphson method. However, the major drawback of the water system simulation method incorporating iterative solution method was its inability in dealing with the rapid growth in the computational requirements with the increase of the physical network size. A classical solution to this would be to partition the system into smaller subsystems, solve the derived subsystems then combine the subsystem solutions to yield the overall network solution. An algorithm which is a generalisation of Kron's idea on tearing linear systems[1] is developed[2]. The nonlinear network tearing algorithm[2] introduces a particularly efficient method of calculating a coordinated solution as it is a direct analytical technique avoiding any iterative recalculation of subsystem solutions. The system is solved as a sequence of Newton_Raphson iterations with each iterative correction calculated as diakoptical coordinated solution to linearized subsystem.

The most important feature of nonlinear diakoptics is that, the form of decomposition resulting from the application of our method maps easily onto parallel processing hardware giving rise to an approximately n-fold improvement in the computational efficiency. In this case the algorithm is to run on a loosely_coupled distributed target platform comprised of a number of SUN workstations networked by an Ethernet communication link. The decision to program the algorithm for loosely_coupled systems was based on the characteristics of the water distribution telemetry system. These tend to grow in a modular fashion by adding new subnetworks to the existing telemetry system, thus significantly augmenting computational load. In such a case, the addition of the new computational node both mirrors the physical system development and is the most economical development of computing resources.

With reference to the mathematical model of the water networks, the main computational effort involved in solving (8) and (9) is the inversion of the Jacobian matrix J. Applying nonlinear diakoptics to the system results in the partitioning of J into blocks representing the jacobian of the subsystems. Solving (8) "block-at-a-time" and then coordinating partial solutions, computational saving can be achieved. However, while the "partition jacobian" approach works well when implemented on a single CPU computer it is not best suited for parallel distributed processing architecture as it tends to overburden the coordinating task with additional calculations and sending large amount of data associated with each jacobian block. Furthermore, sending large amount of data in a distributed system creates large overheads which ultimately result in large processing times. Consequently the parallel processing gains are easily wasted on coordination and communication overheads. To avoid such problems the network is partitioned before its linearization. This means that the subsystem's jacobians are calculated on the individual processing units, which in turn means that a smaller amount of data is transferred between the coordinating unit and the worker processors.

3. Virtual Node Concept

The decomposed model of the water network comprises a coordination level and a subsystem solution level. This model lends itself well to distributed processing.

Implementation of a distributed program in Ada to run on the loosely_coupled processing nodes requires a special mechanism by which the distributed processing tasks can exchange

messages in the course of computations. Although Ada provides an intertask communication mechanism in the form of rendezvous, for the single processor environment, this mechanism requires extension because Ada was not originally designed for distributed systems[5]. This mechanism is the **Ada Virtual Node model** described in[4]. Virtual Nodes can be completely defined in terms of Ada concepts. In the most general case they are the transitive closure of a procedure in an Ada library; the units of such closure must be compliant with a set of "composition rules" which provides the virtual node with the required properties. These properties include: complete encapsulation of internal state, that can only be accessed through a proper interface; the lack of reference to shared objects among virtual nodes, each virtual node must have its own thread of control. Communication between virtual nodes is constrained to happen via "Remote Entry Call"; remote entries are those declared in a task contained in an interface package provided by a virtual node. Such entry calls can be called by other virtual nodes if the interface package is visible to them (by the means of the WITH statement). The communication protocol is then the Ada rendezvous model.

Since virtual nodes are functional abstractions they do not require any immediate mapping onto physical nodes. Therefore the whole application can be designed as a single Ada program made of many communicating virtual nodes, where the synchronization and communication mechanism used is Ada tasking. A Virtual node allocated to a physical node(a processor in the underlying architecture) is transformed into a separate Ada program: the remote entry calls that happen to cross the boundaries of the physical node are translated into calls to primitives provided by a "standard interface" that implement an interprocess communication protocol(e.g Internet_UDP) semantically coherent to the Ada rendezvous on a given execution environment(e.g. UNIX environment). Strong type checking is enforced at the boundaries of the communicating subsystems, and a single interprocess communication scheme with well defined semantics(i.e. the Ada rendezvous) is used.

4. Virtual Node Implementation of the water network simulator

The task of programming the water system simulation algorithm in Ada using VNs is performed in two stages: first the algorithm is segmented along the functional boundaries and defined as VNs, and then VNs are transformed into what is called "Ada Nodes". The VNs to be mapped onto processor units in the underlying execution environment, are given an interface package through which Ada Nodes can communicate with other Ada Nodes in the network.

In our water network simulation program, it was necessary to identify the program components that can be programmed as Virtual Nodes or Virtual Node Types(worker tasks). Since the decomposed model of the system consists of coordination and subsystem solution processes, they can be coded as separate VNs. Therefore, a coordination process can be defined as a VN and subsystem solutions can be constructed from the instances of VN types. VN types represent the subsystem solution worker tasks which can vary in numbers depending on the network's partitioning scheme.

Once all the components are identified and programmed as VNs or instances of VN types, they can be mapped onto the underlying target distributed system. Ideally each unique VN and instances of VN type should map onto a single processor in the target system, however, there may not be as many processors as there are VNs and instances of VN types, thus

Virtual Node grouping is required. VNs are organised into different libraries which are part of the same family of libraries. For example all the public template units(i.e. template-units used by all or some of the virtual nodes (Fig 1)) are grouped together in a single library unit, thus it is important for VNs to have access to this library. **Alsys Multi_library** mechanism provides this facility by allowing references from one library to units of other libraries in the same family of libraries.

The program is implemented on a network of **SUN/SPARC** workstations connected by an Ethernet communication link, having a UNIX operating system. The inter_process communication provided by UNIX is based on **connectionless** User Datagram Protocol(**UDP**) sockets.

The primary task of every worker program is to send its interface task's address which is referenced by an access value to the coordination program. This would then enable the coordination program to call the target worker's interface task. However, sending the access values referencing task objects on another machine is a hazardous task since the use of such parameters in a different address space may have unpredictable effect. Thus it is imperative to restrict the use of such parameters only to the machine where they originated from.

Upon receiving the access value of the worker's interface task, the coordination task incorporates it with the worker's other identification parameters(i.e VN number, and its processor ID number) in a record. In fact there is an array of such records whose elements represent unique identification information about the communicating worker tasks.

In contrast, the coordination interface task prepares the data destined for each worker task. This forms a packet of data incorporated in which is the unique identification information of each worker task. In our implementation the largest packet of data sent once from the coordination VN to the worker VNs, is of the order of 18KBytes for a 130 node water network. However, such large packets would only be sent if the network is partitioned into two subnetworks, which does not typify a normal partitioning criteria of a large network(i.e usually such a large network would be further partitioned into more than two subnetworks).

Prior to receiving the data packet, the worker task creates a receiving buffer which should be as large as the data packet or even larger to avoid buffer overflow. As the size of the data packet depends linearly on the size of the subnetwork, the greater number of partitioning is decreasing the size of the packet while increasing their number. These two factors balance out.

The worker task uses the access value of the worker interface unpacked from the data packet, in order to call the interface task of the target worker. This ensures that the correct instance of the worker VN type receive the data. The worker tasks upon completion of their task(i.e solution of subsystems), organise the results in a packet and send their result packets as soon as they become available. These packets are queued on the coordination VN. This queuing is necessary since the same communication entry point is used by the coordination program to receive the result packets sent by the worker task VN types.

The order in which these packets arrive is arbitrary since each packet has associated with it a software node identification number indicating the worker from which it originated from.

This number is used to unpack and store the parameters of the result packet in the coordination storage area.

Furthermore, the nonlinearity of the water network requires an iterative solution as mentioned in the preceding section, however the amount of data sent in each iteration after the first iteration, would be considerably less because the data packets do not include topological information that does not change from iteration to iteration.

5. Conclusion

This paper presents the application of Virtual Nodes for the distributed implementation of the nonlinear network tearing algorithm[2,3] underlying the water network simulation. The software has been implemented on a network of loosely_coupled SUN/SPARC workstations connected by an Ethernet communication link. The remote communication mechanism(Remote Rendezvous) employed in the VN approach has been tested successfully. The communication overheads associated with the operation of Virtual Nodes have shown to be small and this coupled with the relatively small amount of data transferred between the nodes renders the distributed implementation of diakoptics to be efficient.

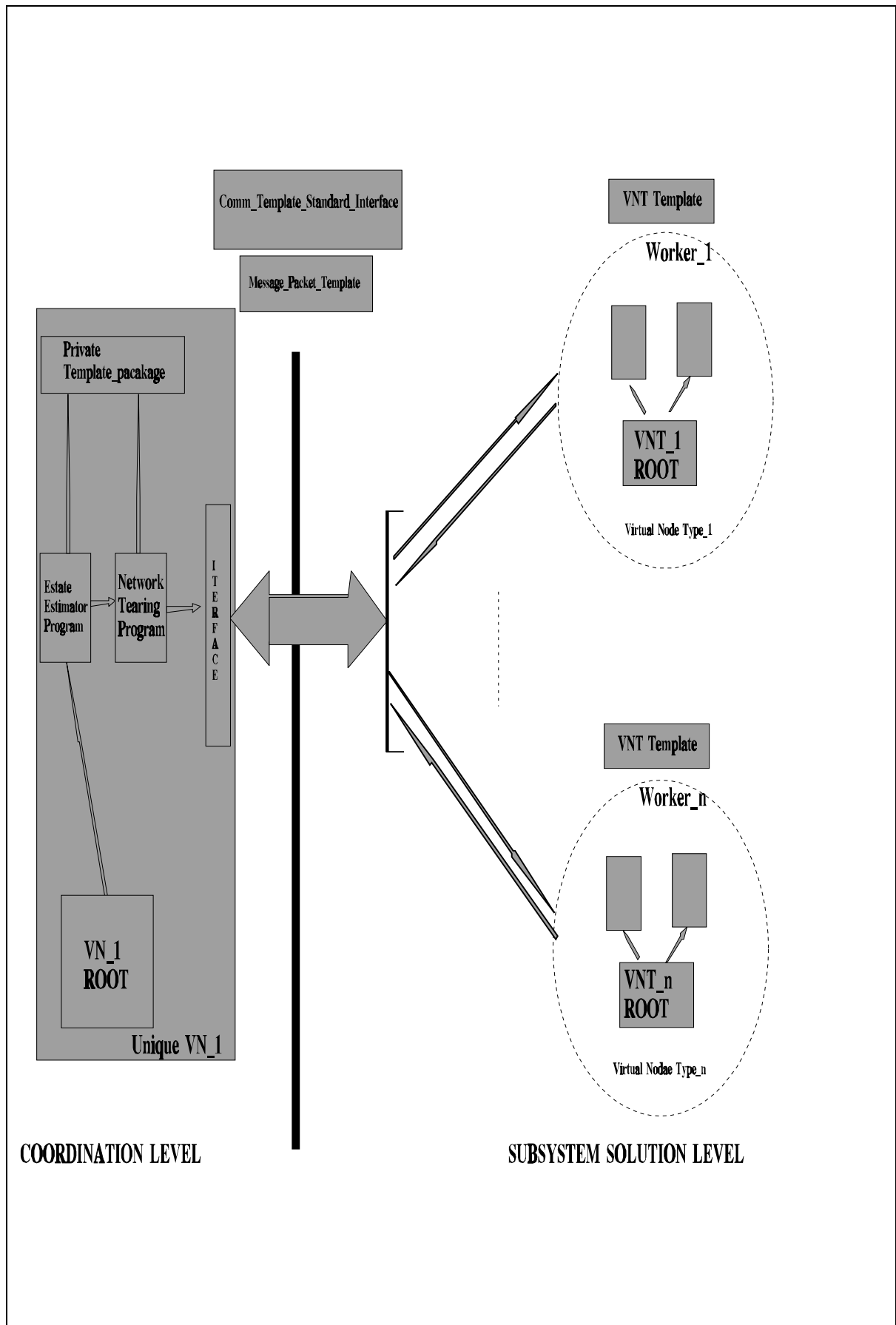


Figure 1: The Virtual Node Construction for Water Network Simulation program.

References

- [1] Kron G., "Diakoptics: The piecewise solution to large scale systems", Macdonald, 1963.
- [2] Hosseinzaman A., Bargiela A., "Parallel simulation of nonlinear networks using diakoptics", Proceedings PACTA'92, Barcelona, Sept.1992.
- [3] Bargiela A., "Nonlinear Network Tearing Algorithm for transputer system implementation", Nottingham Trent University, 1992.
- [4] Atkinson C., Moreton T., & Natali A., "Ada for distributed systems", Cambridge Univ. Press, 1988.
- [5] US Dept of Defense, "Reference Manual for the Ada programming language", ANSI/MIL-SAD, 1815A, 1983.
- [6] Coulbeck B., Sterling M., "Optimised control of water distribution systems", IEE Proc., Vol.125, Oct.1978.
- [7] Sterling M., Bargiela A., "Minimum norm state estimation for computer control of water distribution systems", IEE Proc., Part D, Vol. 131, March 1984, pp.57-63.