International Journal of Quantum Information
Vol. 3, No. 1 (2005) 159–164

World Scientific
www.worldscientific.com

SEMICLASSICAL QUANTUM COMPUTATION SOLUTIONS TO THE COUNT TO INFINITY PROBLEM: A BRIEF DISCUSSION

BURÇ GÖKDEN
Department of Physics, Bilkent University
Ankara 06533, Turkey
gokden@bilkent.edu.tr

Received 21 June 2004

In this paper we briefly define distance vector routing algorithms, their advantages and possible drawbacks. On these possible drawbacks, currently widely used methods split horizon and poisoned reverse are defined and compared. The count to infinity problem is specified and classified to be a halting problem, and a proposition stating that entangled states used in quantum computation can be used to handle this problem is examined. Several solutions to this problem by using entangled states are proposed and a very brief introduction to entangled states is presented.

Keywords: Entanglement; halting problem; counting to infinity problem; routing information protocol; distance vector routing; quantum computers; computer networks.

1. Introduction

Distance vector routing is a popular dynamic routing algorithm which is used in many applications due to its simplicity and ease of implementation. Although it is not a technically superior algorithm, its availability even before it was standardized has made it the most common algorithm used till now. Despite the advantages of the distance vector routing algorithm, which was firstly proposed by Ford and Fulkerson,[1,2] the algorithm has a very critical problem embedded in it, which arises when one of the nodes in the network goes down (or is isolated from the network). Since distance vector routing simply depends on routing table exchange of a node with its neighbors, the other nodes neighboring the node that has just gone down still think that their other neighbors have a better path leading to that isolated node, which in turn starts an endless exchange of data between the nodes, the well known "count to infinity" problem.

A good standard in distance routing protocols is called RIP (Routing Information Protocol)[3] and it solves the count to infinity problem by adding more check actions and limitations to the system; these methods are called split horizon and poisoned reverse. Split horizon together with poisoned reverse solves loops in the network up to and including two gateways, and if more than two gateways are in a

loop, the problem is not eliminated. Poisoned reverse imposes a special meaning on the infinite distance metric (16 for RIP) and updates other nodes' routing tables accordingly to avoid looping so that the neighboring nodes get infinite metric entry into their corresponding tables to immediately prevent a loop. But poisoned reverse has a serious problem in that it limits the bandwidth of the system since the packets that prevent the loop get bigger and bigger as the network neighborhood enlarges. For this reason, RIP is suggested to be implemented in networks no more than 15 hops (i.e. 15 gateways connecting asynchronous networks to each other is an example). Therefore, the major ways of preventing the count to infinity problem lead to more complications and changes in the protocol, which add overheads in using time and space sources (i.e. more delays and less bandwidth with no improvement in performance but some stabilization). Moreover, these preventing algorithms are only applicable when the network size is small, which is due to the time considerations in loop detection. To propose a method for avoiding the count to infinity problem with minimal loss in time and space, we will approach the system as a cause for the problem. The next section will show that the count to infinity problem is a very hard problem (in view of algorithmic complexity considerations) to be efficiently avoided by classical computers if the case is imminent in a certain algorithm.

2. Reduction of Counting to Infinity Problem to the Halting Problem

The halting problem is one of the oldest unsolvable problems of computation theory. It stems from Hilbert's Entscheidungs problem (decision problem) which asks whether there is an algorithm for the solution of any given problem. The halting problem is Turing's answer to this question and it is a well known issue that, unfortunately, all problems do not have feasible algorithms for their solution. The halting problem can be stated as follows[a]: there exists no procedure $h(x)$ that determines whether a Turing Machine M_x will halt upon input of a specific string x. In other words, no algorithm can tell whether another procedure would halt or loop forever.

The count to infinity problem can also be defined in the class of problems that have the characteristics of the halting problem, and to show this it is enough to show a function derived from the count to infinity problem which models the function $h(x)$ above. Our $h(x)$ for the count to infinity problem can be given as

$$h(x) = \begin{cases} 0, & \text{if there is not a well-defined} \\ & \text{route to a just isolated node with an} \\ & \text{acceptable amount of metric used,} \\ 1, & \text{if there is a well-defined route} \\ & \text{to a just isolated node with some optimum} \\ & \text{finite amount of metric used.} \end{cases}$$

[a]A proof can be found in Ref. 4.

Here, the problem only covers the nodes that are on the path that is not available just after the destination node is isolated. The halting problem is an open-ended issue in computer science that implies no current solution with classical computers. When one attempts to solve this problem based on a classical computer architecture, one must either completely change the algorithm or take some precautions not to let it happen, which in turn adds significant overheads and limitations to the system. Split horizon with poisoned reverse is such a precaution, which still has problems but works well in small networks to some extent. For these reasons we will propose a new feature from quantum theory, entangled states, to prevent this problem for larger networks, which change the hardware by using a phenomenon that has no classical counterpart in physics. The main aim of this change in hardware is to design a feasible system that extends to distant networks with minimum overhead, to significantly reduce the time and space complexity of the system which is practically applicable by the users of the network.

3. Application of Entanglement to the Counting to Infinity Problem

Entanglement may be applied in many different ways to a classical network. We will now study these applications, and then compare them with each other. During these comparisons, we will assume that the entangled states can be transported via a quantum channel so that nodes can share them.

We will fix the following convention on the communication via qubits. Assume that two nodes A and B in a network have corresponding entangled pairs which are represented by the state $\frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle)$ with them. If the sender A detects that it is completely isolated from the network it does a measurement on its qubit by the projection operator $P_0 = |0\rangle\langle 0|$. When the receiver B sees that no information is coming via classical channels it applies a projection operation of $P_1 = |1\rangle\langle 1|$ on its entangled state. If the node A is isolated (i.e. the state has the form $|0\rangle|0\rangle$ after the projection), node B will get zero output as a result of its measurement. If A is still connected to the network, it does no operation on the system and B's measurement projects the system onto state $|1\rangle|1\rangle$.

One application may be implementing entangled states (there may be a lot of pairs shared between two nodes) between neighboring nodes of a selected node A and distant gateways. When that node A goes down, a neighboring node learns it and then sets its entangled state accordingly and the distant gateway periodically measures its entangled state. Since the count to infinity procedure has already begun near the node A, this gateway is aware of the situation and it may guarantee that the problem does not pass through it to outer networks and stays in a limited area so that if split horizon with poisoned reverse is still active, the resolution of the problem becomes faster. This can also help the distance vector routing to be used in much larger networks since the count to infinity problem is guaranteed to be limited in a smaller area if it occurs and split horizon with poisoned reverse

can handle it up to some acceptable level. Here, infinity will be represented by the number of hops to reach the distant gateway that knows the situation at node A plus 1. This system requires periodic transportation of entangled pairs since the distant gateway periodically consumes the states it has by measuring them to learn whether node A has detected any problems around itself. This scenario is so simple but even in this case the complexity of informing a distant node about a change in topology becomes $O(1)$. Variants of this type of informing a change in topology may be continuously applied among nodes but this also adds an overhead of qubit exchange on the network. However, this exchange does not significantly affect the performance since we only send 0 or 1 data and qubits are independent of each other due to the fact that each of them is discretely entangled, and this makes the quantum channel error tolerant, i.e. if one or two qubits are mistakenly measured on the way, they become useless but this does not affect the other qubits and this is not a significant source of unreliability.

Another application that is more sophisticated is the exchange of entangled qubits generated at a node while exchanging the distance vector routing data between neighbors. When one node updates its table, it also gets corresponding entangled qubits generated by its neighbors. Each node i generates and stores n_i pairs of qubits for each of the N possible destinations and for itself (since it may be the destination of one of its neighbors), where n_i is the number of neighbors of node i. Thus, each node stores $(N+1) \times n_i$ qubits at most because for a specific destination every neighbor of node i may pass through it in the worst case. If the neighbor of node i decides to pass through it, node i also sends the one of the reserved entangled qubits for the destination. This procedure can be explained on a simple network lined up on a straight line which has the morphology

$$A----B----C------D----E--\cdots$$

The process for the case of node A can be described as follows:

(i) Node A goes down and measures qubits reserved for itself;
(ii) B could not get a distance vector[b] from node A and measures the qubit in the entry for node A as the destination in its routing table and learns that A is offline.
(iii) B checks its neighbor C and sees that there is a path from C to A, without knowing that this path passes through it (counting to infinity just starts).
(iv) Since B knows that it cannot reach A with its former path, it measures all the n_B qubits stored for node A as the destination.
(v) C sees that all its neighbors are three hops away from the destination node and it measures the qubit reserved for node A in its routing table, which still defines its former path through B and sees that A is not reachable through B. Then it measures n_C qubits for destination A and selects D as a path.

[b]More information on distance vector routing can be found in Refs. 3 and 7.

(vi) In a second exchange to update its routing table, B measures the qubit in its entry for node A (this time the qubit has come from neighbor C) and learns that A is not reachable through C and concludes that it cannot reach A in any way.

(vii) After several exchanges, the system eventually updates itself correctly. In the above network, if D is the last node in the linear chain, D's update concludes that none of the nodes are reachable.

Thus the system quickly and asynchronously (i.e. no periodic measurement of qubits; they are measured just before each exchange of routing data and generated and exchanged while exchanging the routing data) collapses to a stable system with its new network topology. The protocol defined above can be applied on more complex networks than the above example and it prevents the count to infinity problem in an efficient way when it occurs.

4. Conclusions

In this paper we briefly defined the distance vector routing algorithm characteristics and examined possible causes and results of the count to infinity problem. We investigated the advantages, disadvantages and limitations of the most used methods to avoid the count to infinity problem or recover from it. These methods are split horizon and poisoned reverse which can also be used together. The reduction of the count to infinity problem to the halting problem lets us examine the problem from the point of view of computation complexities and a decision problem. We proposed to change the hardware to prevent counting to infinity in order to have a more robust network algorithm without significantly increasing the complexity of the distance vector routing algorithm. For this reason a statistical and nondeterministic theory of computation, quantum computation theory, is used to develop such algorithms. The novel states called entangled states enable the network nodes to communicate with each other without depending on the network connectivity in the case of network topology change, which is possible if a measurement protocol of states is established. Moreover, the time complexity of learning any change (or equivalently, making a measurement on an entangled state) between any node with any amount of distance between them is $O(1)$ in the case of a network topology change, which significantly increases the size and quality of service of the network on which distance vector routing can be applied.

References

1. R. E. Bellman, *Dynamic Programming* (Princeton University Press, 1957).
2. L. R. Ford and D. R. Fulkerson, *Flows in Networks* (Princeton University Press, Princeton, 1962).
3. C. Hedrick, *Request for comments: 1058*, can be found from web page www.ietf.org/rfc (1988).

4. M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2000).
5. P. W. Shor, *SIAM J. Computing* **26**, 1484 (1997).
6. D. Deutsch, in *Proc. Royal Society* **A400**, 97 (1985).
7. A. S. Tanenbaum, *Computer Networks*, 4th edn. (Pearson Education International, 2003), Chap. 5, p. 357.