Network Court Protocol and Malicious Node Conviction

Na Li and David Lee
Department of Computer Science and Engineering, The Ohio State University
lina,lee@cse.ohio-state.edu

Abstract – A Network Court Protocol is designed for malicious node conviction based on information from network node accusing and testifying operations, which are formally modeled by algebraic operators. It is shown that the malicious node conviction is equivalent to the uniqueness of the solution of a system of Boolean equations and that is equivalent to the uniqueness of a corresponding satisfiability problem. A linear time algorithm is presented for the conviction process using a graph search.

I. INTRODUCTION

Malicious nodes disrupt network operations and pose a serious security threat to network applications. They can be detected by their abnormal behaviors, such as generating abnormal traffic [L04]. Consequently, based on the observed behaviors, a node can accuse another one as being malicious or testify for its integrity. On the other hand, such statements by malicious nodes can be false. Our goal is to identify the malicious nodes based on the statements, which are made by good or malicious nodes and can be true or false.

This is similar to court activities. Defendants, prosecutors, and witnesses can accuse or testify for another person. Based on the statements, the jury or judge has to reach a verdict to convict a criminal, or to prove his innocence, or to quit the case for lack of evidence.

This statement-based decision making process is not attack detection or network element authentication [K02]. It is different than Byzantine agreement [L96] and is not a reputation system [R00] either. We call it a *Network Court Protocol* for convicting malicious nodes.

In general, this is an unsolvable problem. For instance, if there are only two nodes who accuse each other, there is no way one can tell whether both are malicious or one is good and the other is malicious. The complication is due to the fact that a malicious node can falsely accuse good nodes, or testify for other malicious nodes, or accuse another malicious node to make the conviction harder, or may even behave non-deterministically. We make the following natural assumptions:

Assumption 1. A good node makes a statement on another node:

(1) Testifying that it is a good node; the testified node must also be good; and

(2) Accusing that it is a bad node; the accused node must be bad.

Remark 1. We do not make any assumptions on the consequences of bad node accusation and testimony, since they can be false.

II. MATHEMATICAL MODEL

For simplicity, we call a malicious node a bad node, denoted by a Boolean value 0, and call a healthy node a good node, denoted by 1.We use operator $\otimes$ and $\oplus$ to denote accusation and testify-for, respectively; $x \otimes y$ for node y accusing node x, and $x \oplus y$ for node y testifying for node x.

By Assumption 1, we know that:

(1) A node must be bad if it is accused by a good node, it is good if it is testified by a good node, thus we have: $*\otimes 1=0$ and $*\oplus 1=1$ where * stands for 0 or 1.

(2) A node does not change its status if it is accused or testified by a bad node, thus we have: $*\otimes 0=*$ and $*\oplus 0=*$.

It can be easily shown:

$$x \otimes y = x \cdot y' \quad (1)$$

$$x \oplus y = x + y \quad (2)$$

where " . " is the Boolean "and" operator that is often omitted, "+" is the "or" operator, and " ' " represents the complement.

Thus the problem can be defined as a general problem in distributed computing as follows; its application to networking is a special case. Here Boolean value 0 and 1 represent bad and good element, respectively.

Court Conviction Problem

Given a set of n elements of unknown Boolean values x_i, $i=1,\ldots,n$, a set of u accusation statements: $x_{i_p} \otimes x_{j_p}$, $p=1,\ldots,u$, and a set of v testify-for statements: $x_{i_q} \oplus x_{j_q}$, $q=1,\ldots,v$, where $\otimes$ and $\oplus$ are the accusation and testify-for operator, respectively, determine the Boolean value of each element.

The problem is *completely solvable* if the value assignment to each element is unique.

From Eq. (1) and (2), we can rewrite the statements in Court Conviction Problem as a system of $u+v$ equations of n Boolean variables:

$$x_{i_p} = x_{i_p} x_{j_p}', \quad p=1,\ldots,u \quad (3)$$

$$x_{i_q} = x_{i_q} + x_{j_q}, \quad q=1,\ldots,v \quad (4)$$

The Court Conviction Problem is now reduced to solving a system of Boolean equations. It has at least one solution: the *true* value of the nodes. The problem is *completely solvable* if and only if the system has a unique solution that is the same as the true values of all the elements.

The above system of Boolean equations is also equivalent to the following [L07]:

$$\prod_{p=1,q=1}^{p=u,q=v} (x_{i_p}' + x_{j_p}')(x_{i_q} + x_{j_q}') = 1 \quad (5)$$

Obviously, it is a satisfiability problem (SAT). We conclude:

Proposition 1. The system of Boolean equations (3) and (4) has a solution if and only if the Boolean expression on the left side of (5) is satisfiable, and the system in (3) and (4) has a unique solution if and only the Boolean expression in (5) is uniquely satisfiable.

Corollary 1. The Court Conviction Problem is completely solvable if and only if the system of Boolean equations (3) and (4) has a unique solution, and this is the case if and only if the Boolean expression in (5) is uniquely satisfiable.

III. NETWORK COURT CONVICTION PROTOCOL DESIGN

The Court Conviction Problem is now reduced to the satisfiability problem and its unique solution. We present a linear time algorithm using a *conviction graph.*

From a Conviction Problem defined in (5), construct a conviction graph as follows. Each accusation *a*-edge from node y to x, $y \xrightarrow{a} x$ is associated with a clause of the Boolean expression in (5): $x' + y'$, which must have value 1 in the conjunctive normal form of (5). Similarly, each testify-for *t*-edge $y \xrightarrow{t} x$ corresponds to a clause $x + y'$ in (5) with value 1.

It can be easily checked that $x + y' = 1$ and $x' + y = 1$ lead to $x = y$. This introduces an equivalence relation (testifying for each other) on the nodes and we can first merge the equivalence classes of nodes in the graph.

Obviously, the Conviction Problem has a trivial solution: all the nodes are bad (all variables have value 0). We rule out this uninteresting case and assume that there is at least one good node. We search the conviction graph in two phases as in Algorithm 1. For detailed explanations of the algorithm, see [L07].

Algorithm 1 (Conviction Graph Search)

Input: *Conviction Graph G= (V, E)*

Output*: Conviction Problem has a unique solution or not. An output array A[u], u=1, 2…, n=|V|, contains node value assignments: 1, 0, or β for good, bad and undetermined node, respectively.*

Initialization:

Queue $Q \leftarrow \varnothing$; /* nodes with determined value 1 */

List $L \leftarrow \varnothing$; /* nodes with unknown value */

for u=1 to n

  if node u is known to be good a priori

    A[u] = 1, ENQUEUE (u, Q);

  else /* node u value unknown */

    A[u] = β, insert u to L;

Search 1:

  while $Q \neq \varnothing$

    u=DEQUEUE(Q);

    for each $v \in Adj(u)$ /* end node of edge from u */

      if A[v] = β

        if E(u, v)='a' /* a-type edge */

          A[v] = 0; /* a bad node */

        else /* t-type edge E(u, v)='t' */

          A[v] =1; ENQUEUE(v, Q) ; /* a good node */

        delete v from L;

  if $L = \phi$ /* search completed */

    return "Conviction Problem has a unique solution."

  else /* search *residual* graph of β-nodes in L */

Search 2:

    Examine t-edges only in residual graph, merge SCCs, conduct a topological sort, obtain a DAG with β-nodes listed in L' in topologically sorted order;

    While $L' \neq \varnothing$

      u=DELIST (L') /* examine β-nodes bottom up in topological order */

      for each $v \in Adj(u)$ /* end node of edge from u */

        if (E(u, v) = 'a' and A[v]=1) or

          (E(u, v) = 't' and A[v]=0)

          u=0, exit; /* cannot assign value 1 to u */

      u=1; /* node u can be assigned value 1 */

      return "Conviction Problem has no unique solution."

    return "Conviction Problem has a unique solution."

Remark 2. The Conviction Problem is completely solvable if and only if Algorithm 1 returns "Conviction Problem has a unique solution." In this case, all the good and bad nodes are uniquely identified: A[v] =1: v is a good node; and A[v] =0: v is a bad node, v=1, 2,…, n.

IV. CONCLUSION

In this paper, we formulate the Network Malicious Node Conviction Problem as a general problem in distributed computing. We propose Network Court Protocol to convict malicious nodes that is applicable to both centralized and distributed networks.

The conviction problem is completely solvable if and only if algorithm 1 returns a unique assignment for all the nodes. However, after Search 1, a set of nodes are uniquely identified and only the nodes in the residual graph may not be determined. As a result, we obtain a partial solution of the conviction problem – we identify some of the malicious nodes from this process.

Our model of node behaviors is rather restrictive and deterministic (Assumption 1). Often in practice accusation and testimony of nodes are associated with certain probability distributions. A general model of network node behaviors and malicious node conviction remain to be investigated.

References

[K02]C. Kaufman et al. Network Security: Private Communication in a Public World. 2002. ISBN:0-13-046019-2

[L04]A. Lakhina et al. Characterization of network-wide anomalies in traffic flows. In Proceedings of the 4th ACM SIGCOMM Conference on internet Measurement, Taormina, Sicily, Italy, 2004.

[L07]N. Li and D. Lee. Network Court Protocol and Malicious Node Conviction. Tech. Report, The Ohio State University, 2007. Reference Number: OSU-CISRC-9/07-TR66.

[L96]N. A. Lynch. Distributed Algorithms. 1996. ISBN:1-55860-348-4

[R00]P. Resnick et al. Reputation Systems. Communications of the ACM, 43(12), pp. 45-48, 2000.