Brief Announcement: Strong Detection of Misconfigurations

Raj Kumar Rajendran
raj@ee.columbia.edu

Vishal Misra
misra@cs.columbia.edu

Dan Rubenstein
danr@ee.columbia.edu

Categories and Subject Descriptors: C.2.2 [Computer Communication Networks]: Network Protocols
General Terms: Algorithms, Security, Theory
Keywords: Distance-Vector, Routing, Anomaly Detection

1. DETECTING MISCONFIGURATIONS

Distributed routing protocols rely on all nodes correctly implementing the protocol if the "best" routes are to be chosen. However, some nodes may mis-implement the protocol. Recent research has identified properties of its routing state that an individual node can check to identify errors elsewhere. A shortcoming of such work is that checking the suggested properties may fail to detect a misconfiguration that could have been identified by checking alternate properties of the state. Our work introduces *strong detection* as a solution. If a node fails to detect a misconfiguration using strong-detection it can be sure that no method exists which can detect the misconfiguration. We present an $O(|V|^3)$ algorithm that implements strong detection for the Distance Vector routing protocol.

1.1 Weak Detection

Consider a network $G = (V, E, W)$ running a routing protocol P, where a node n maintains a protocol-specific *state*, d_n. For instance in Distance-Vector the state d_n is a $|V|$ x $|N(n)|$ table where $d_n(i, j)$ is the shortest path distance neighbor n_j claims from itself to node i. Suppose that node $\hat{n}$ reports erroneous distances leading other nodes to select non-optimal routes.

It has been shown that the triangle inequality can be applied to Distance Vector to detect misconfigurations[2]. A problem with such a method is that if a violation of the specific property is identified, then clearly a misconfiguration exists. However when a violation is *not* found, it does not necessarily mean that a misconfiguration does *not* exist. Checking for a different, but unknown property may have revealed the misconfiguration. Methods such as these provide what we call *weak detection* where, given a node's state, a misconfiguration is not detected either because the misconfiguration is undetectable or because the misconfiguration is detectable by checking some property, but the weak detection method did not check the correct property.

To see the drawback of weak detection, consider a network with four nodes A, B, C, N where only edges of weight 1 or 2 are allowed and A and B are N's neighbors and are connected to it by edges of weight 1. A and B report to node N that their shortest-path distances to nodes A,B, and C are 0,2,2 and 2,0,3 respectively. It can be verified in this example that while the triangle inequality is not violated in node N's state, a misconfiguration must in fact exist because no valid graph can be constructed that would yield N's state (details are provided in [1]). Here, the weak detection method of checking the triangle inequality property did not identify the misconfiguration.

PODC'05, July 17–20, 2005, Las Vegas, Nevada, USA.
ACM 1-59593-994-2/05/0007.

1.2 Strong Detection

Our work investigates what we call *strong detection* which *must* detect any misconfiguration that is detectable by checking *any* property. Given d_n the state at node n it works as follows: for each graph $G \in \mathcal{G}$ where $\mathcal{G}$ is the set of all *valid* graphs, the protocol P is simulated atop G generating the states that would be stored in each node in G. If no graph $G \in \mathcal{G}$ exists for which P simulated on G produces the state d_n for node n, then a misconfiguration must exist. This is because *some* graph in $\mathcal{G}$ must accurately describe the network, so if simulating the protocol P on every $G \in \mathcal{G}$ does not produce d_n, only a mis-computation somewhere in the distributed running of the protocol would yield state d_n at n. If on the other hand some valid graph $G \in \mathcal{G}$ exists such that running P on G produces state d_n it is possible that G is the actual graph, or that the actual graph is some other graph $\hat{G}$ whose correct state $\hat{d}_n$ does not match d_n, but a misconfiguration yielded d_n at n. However, given only the state information d_n, there clearly is no way for node n to determine whether d_n was derived correctly for a graph such as G, or incorrectly for a graph such as $\hat{G}$. The challenge of course, is to find a way to explore all possible graphs $G \in \mathcal{G}$ in a reasonable amount of time.

We have successfully identified an efficient ($O(|V|^3)$) procedure that performs strong detection for the Distance Vector routing protocol. A node n which knows its state d_n performs the following procedure: 1. n executes the $O(|V|^3)$ algorithm given below that outputs from d_n a particular *canonical graph* G'. 2. If G' is not valid ($G' \notin \mathcal{G}$), then a misconfiguration must exist. 3. If G' is valid ($G' \in \mathcal{G}$), then n simulates Distance Vector on G', producing simulated state d'_n for node n. 4. If $d'_n = d_n$, then we have identified a valid graph G', and so there is either no misconfiguration or it is impossible to detect, since G' would have produced such state. 5. If $d'_n \neq d_n$, then there is no graph $G \in \mathcal{G}$ that would produce state d_n when distance vector is run on it. This is a rather strong claim and so the proof is provided in [1].

The canonical graph G' is formed by connecting every pair of nodes i, j in the graph with an edge whose weight $w'(i, j)$ is the smallest value in $S_{i,j}$ that is no less than $\max_{k \in N(n)} |d_n(k, i) - d_n(k, j)|$. If no such value exists (in the case where this maximum is larger than any value in $S_{i,j}$), then the edge is omitted. After all edges are constructed, if $G' \in \mathcal{G}$, the distance vector algorithm is simulated on G' producing a state table d'_n for node n. Then d'_n is compared to the original state table d_n within which we are attempting to identify a misconfiguration. The theorem is presented below. The reader is referred to [1] for the proof.

THEOREM 1.1. *d_n is a valid state table for some graph $G \in \mathcal{G}$ if and only if it is valid for the canonical graph, $G' \in \mathcal{G}$.*

2. REFERENCES

[1] Raj Kumar Rajendran, Vishal Misra, Dan Rubenstein, "Strong Detection of misconfiguration," http://www.ee.columbia.edu/ kumar/papers/podc05.pdf.
[2] Dan Pei, Dan Massey, and Lixia Zhang, "Detection of invalid routing announcements in rip protocol," in *Proc. of IEEE Globecom, San Francisco, CA*, December 2003.