A Formal Specification for RIP Protocol

Dan Pei[1], Dan Massey[2] and Lixia Zhang[1]

UCLA[1]
{peidan, lixia}@cs.ucla.edu

Colorado State University[2]
massey@cs.colostate.edu

UCLA CSD Technical Report TR040046

Abstract

A protocol standard is often intended to allow multiple implementations to inter-operate, and multiple implementation choices and many engineering details usually make a formal protocol specification difficult. Lack of formal protocol specification has two important results, as has been shown in the IETF standard development process: the correctness of the protocol is not easy to be proven; the protocol may be ambiguous in some aspects, leaving rooms for implementation bugs and even for attacks. Even worse, the bugs and ambiguities are identified in an ad hoc way, and there has not been any systematic way to identify bugs and ambiguities in existing protocols.

In this work, we present a formal specification for the Routing Information Protocol(RIP). In Section 2, we will give a formal specification of the minimal requirements for a RIP router in order to guarantee that RIP will converge after a network topology change. By analyzing the RIP standards, we only specify those requirements that must be satisfied, while leaving room for any implementation choices allowed. Then in Section 3, we will present another formal specification of RIP by Finite State Machine. Using FSMs, we are able to find two ambiguities in the RIP standard.

I. Introduction

Designing network protocol is still an engineering art nowadays. This is especially the case for routing protocols. Based on some distributed algorithms(distance vector, path vector, or link state), protocol designers specify the protocol by stating each entity(router, area, or AS) in the network should do using language description, pseudo-code, or some other tools such as FSM(finite state machine). There are works that proved the correctness of the distributed algorithms behind the routing protocols, which usually have formal specifications. However, a protocol standard is intended to allow multiple implementations to inter-operate. Therefore, a protocol standard should summarize the operations of every possible implementations. Multiple implementation choices and many engineering details usually make the formal specification difficult.

Lack of formal specification has two important results, as has been shown in the IETF standard development process: the correctness of the protocol is not easy to be proven; the protocol may be ambiguous in some aspects. First, bugs in the protocol specifications have been continuously identified and then fixed. Second, the protocol ambiguities also leave room for implementation bugs and even for attacks. These bugs and ambiguities are identified in an ad hoc way. Third, protocols may be used in practice without the correctness proof. For example, the de facto inter-domain routing protocol BGP may not convergence even if its protocol standard[9] is strictly followed[10], [6].

In this work, we presents a formal specification for the Routing Information Protocol(RIP)[7], which is probably the simplest among existing routing protocols. In Section 2, we will give a formal specification of the minimal requirements for a RIP router in order to guarantee that RIP will converge after a network topology change. By analyzing the RIP standards[7], we only specify those requirements that must be satisfied, while leaving room for any implementation choices allowed. Then in Section 3, we will present another formal specification of RIP by Finite State Machine. The RIP FSM specification gave a reference model for implementation which covers what a RIP router MUST do and what a RIP router SHOULD do. The meaning of MUST and SHOULD is interpreted as described in RFC 2119[4]. Using FSMs, we are able to find two ambiguities in the RIP standard.

In the past, [1] provided a formal specification for distributed asynchronous Bellman-Ford Algorithm, and a proof that assuming there is no change in the network topology, this algorithm eventually converge to a state

where each route has the correct shortest distance to the destination. Using HOL and SPIN, [2] provided a proof of convergence of simplified RIP. However, the formal specification used during their proof doesn't consider the levels of requirements. For example, according to RFC 2453[7], a RIP router MUST implement simple split horizon and SHOULD implement split horizon with poison reverse. But in their formal specification, every RIP router implements split horizon with poison reverse. In this sense, both the formal specification and proof are incomplete.

II. Formalization of RIP

A. RIP protocol

Each RIP router maintains a routing table. There is one routing entry for each destination:

- the shortest distance from the router to the destination.
- next router along the path to the destination.

Routers periodically advertise their distance to the destination to their neighbors. Upon receiving an advertisement, the router checks whether any of the advertised routes can be used to improve current routes. Whenever this is the case, the router updates its current route to go through the advertising neighbor. Routes are compared by their length. The value of route length must be an integer between 1 and 16, where 16 means infinity.

In order to avoid loops, routers MUST do split horizon when advertise the route to the neighbor where the route is learned from: either omit the route, or set the distance to be infinity.

Every 30 seconds, routers advertise the complete routing table to every neighbor router. For each route, there are two timers, a "timeout" and and a "garbage-collection" time. If a route hasn't been refreshed for 180 seconds, the router will assume that there was a link failure, the destination will be marked as unreachable and "the garbage-collection" be set to 120 seconds. If the timer expires before the entry updated, the route is expunged from the table.

B. Minimum Requirements

For one single router r that has n neighbors, given a set of neighbors $Neighbor(r)$, and a fixed set of distances from r to each neighbor $i \in Neighbor(r)$: $C_r(i)(C_r(i) \geq 1)$, denotes the distance from r to neighbor i.

For the destination that is directly connected to router r, the distance is a fixed one. So we only consider the route to one destination d that is not directly connected to router r.

Let:

- T= Current time
- $M_r(i)$ =distance last learned from i prior to T.
- $P_r(i)$ = Time when the last distance was learned from i prior to T.
- L_r = neighbor whose distance was last learned prior to T.
- $D_r(i,T) = min(16, M_r(i) + C_r(i))$.

Note that if $T - P_r(i) = 180$ seconds, $M_r(i)$ will be expired, and we treat such an expiration as a new distance(16) learned from neighbor i. The definition of $M_r(i)$, L_r,and $D_r(i,T)$ also apply to these specially learned distances.

If a RIP router has routing entry for destination d, the router must always maintain $Dist_r(T)$ and $Nexthop_r(T)$ such that:

R 1: $If \quad Dist_r(T) < 16, then \quad Dist_r(T) = D_r(Nexthop_r(T), T) \quad AND \quad Nexthop_r(T) \in Neighbor(r)\}$

R 2: $Dist_r(T) <= D_r(L_r, T)$;

R 3: $If \quad Dist_r(T) > Dist_r(T'), then \quad Nexthop_r(T') = L_r$

R 4: $\forall i \in Neighbor(r)$, a working RIP router r must send at least one update containing $Dist_r$ for d to i every 180 seconds unless $Nexthop_r = i$.

(R1), (R2), (R3) and (R4) give a high-level and formal specification of the minimal requirements for a RIP router, while leaving room for any implementation choices allowable by the RIP standard. They are elaborated in the following.

- (R1) can be viewed as the route validity requirement: satisfying this requirement is sufficient and necessary to guarantee that only a route last advertised(through a real update) by a neighbor may be chosen as the best route, and that the route expires if not refreshed within last 180 seconds(through specially learned distances).
- (R2) requires that the best distance selected must be no more than the last distance learned. This must be true in order to select the best route among its neighbor's most recently advertised routes available to a RIP router. The exact meaning of "available" depends on the actual implementation. For example, a RIP router usually only keeps the best route, the most recently advertised routes "available" to a RIP router are the current best route and the route just received if they are from different neighbors; in this case, $Dist_r(T) = Min\{Dist_r(T'), D_r(L_r, T)\} \leq D_r(L_r, T)$. If they are from the same neighbor, the only most recently advertised route is only the route just received; in this case, $Dist_r(T) = D_r(L_r, T)$. It is also possible that the RIP router only knows the best route and the fact that it has just expired; in this case,$Dist_r(T) = D_r(L_r, T) = 16$. In all of these cases, a RIP route must satisfy (R2). Although only keeping the best route is the most common implementation, the RIP standard[7] allows other choices. If a RIP router keeps all of its neighbors' most recently advertised routes(as in Asynchronous Bellman-Ford Algorithm), the most recently advertised routes "available" to the RIP router are the route just received and the routes most recently advertised by other neighbors. It is also possible that in some implementations, a RIP router keeps some of its neighbors' recently advertised routes. No matter what the implementation is,(R2) must be satisfied.
  (R2) guarantees that when the shortest route arrived to a RIP router, it will be chosen as the best route.
- (R3) summarizes what is necessary when $Dist_T$ may increase according to [7]: the current best route expires or a longer distance received from the current nexthop. However, in different implementations, $Dist_T$ may increase in different ways. If the RIP router only keeps the best route, $Dist_T$ just increase to $D(Nexthop_r(T), T)$; if the RIP router keeps all of its neighbors' most recently advertised routes, $Dist_r(T)$ may increase to some value between $Dist_r(T')$ and $D(Nexthop_r(T), T)$ inclusive.
- (R4) If router r fails to satisfy (R4) to neighbor i, i will assume that r either crashes or loses the route to d.

The minimum requirements (R1), (R2), (R3) and (R4) are sufficient to guarantee that RIP always converges. Each of them is required by RIP protocol, and any combination of 3 of them is not sufficient for RIP to convergence. Proof is provided in Appendix.

III. State Machine for route to p

Our first effort to precisely specify RIP is to use the Finite State Machine tool. In this section, we will give a state machine which specifies the details of RIP operation on one route which is required and recommended by RIP standard[7]. Given that the RIP standard[7] only provides detailed requirement for the most common implementation choice that chooses to only keep the best route, we only specifies the details for this implementation choice.

Details of transitions are obtained by following the requirement of (R2), (R1), (R3), (R4) and RFC[7]. The state machine , together with the output state machine in the next section, shows a RIP implementation whose correctness can be proved by exhaustively verifying that (R2), (R1), (R3), and (R4) are always satisfied.

To define a finite state machine, we should do the following according to the protocol specification.

- identify the data structure to work on
- identify all the external events
- identify all the internal timer events
- identify all the actions on the data structures and timers
- identify all the outputs
- define a set of states, and specify one Start State.

By identifying all of these from protocol specification, we will be able to define a state machine for one data structure and then claim it describes the protocol precisely. Furthermore, we can also find the ambiguities if there is any event whose transition is not specified by the protocol specification.

We found it is appropriate to model the RIP protocol using two state machines: one for one routing entry (the route to one destination), and one for output process. The reason is that each routing entry are treated individually, but output process basically works on the whole routing table. The two state machines are communicating state machines[5], in the sense that they both work on the routing table.

Data Structure:

- one route entry in the routing table(RIB)
- one entry in FIB(forwarding information base)

External Events:

- Update(N,D): new update from neighbor N, metric==D

Timers Events

- route timeout(180-second) timer expires;
- garbage-collection timer(120-second) expires;

Actions on data structure and timers

- restart 180-second timer;
- restart 120-second timer;
- clear 120-second timer.
- Update RIB;
- Update FIB;

Output:none.

Start State:NO_ROUTE

Proof: Exhaustive verification shows that the state machine shown in Figure 1 always satisfies (R2), (R1), (R3), and (R4) are always satisfied.

IV. State Machine for the RIP Output Process

Data Structure

- Routing Table(Read-only except the change flags)

External Events:

- Routing table changed;
- request from neighbor i;

Timer Events:

- 30-second timer expires;
- triggered update timer expires;

Actions on data structure and timers

- clear the change flags in all entries in the routing table;
- restart triggered update timer;
- restart 30-second timer;
- clear triggered update timer;

Outputs:

- send out update to all neighbors including whole table;
- send out update to all neighbors including changed entries;
- send out update to one particular neighbor;

Start State:triggered update expired

Ambiguities identified through developing state machines

1) ”triggered update timer counting down(routing table changed)” state receives event ”30-second timer expires”, RIP standard does not specify what to do with the triggered update timer: clear it, restart it or leave it as it is? Although it may not affect RIP operation significantly, it is an example that protocol ambiguity could be identified through developing FSMs.

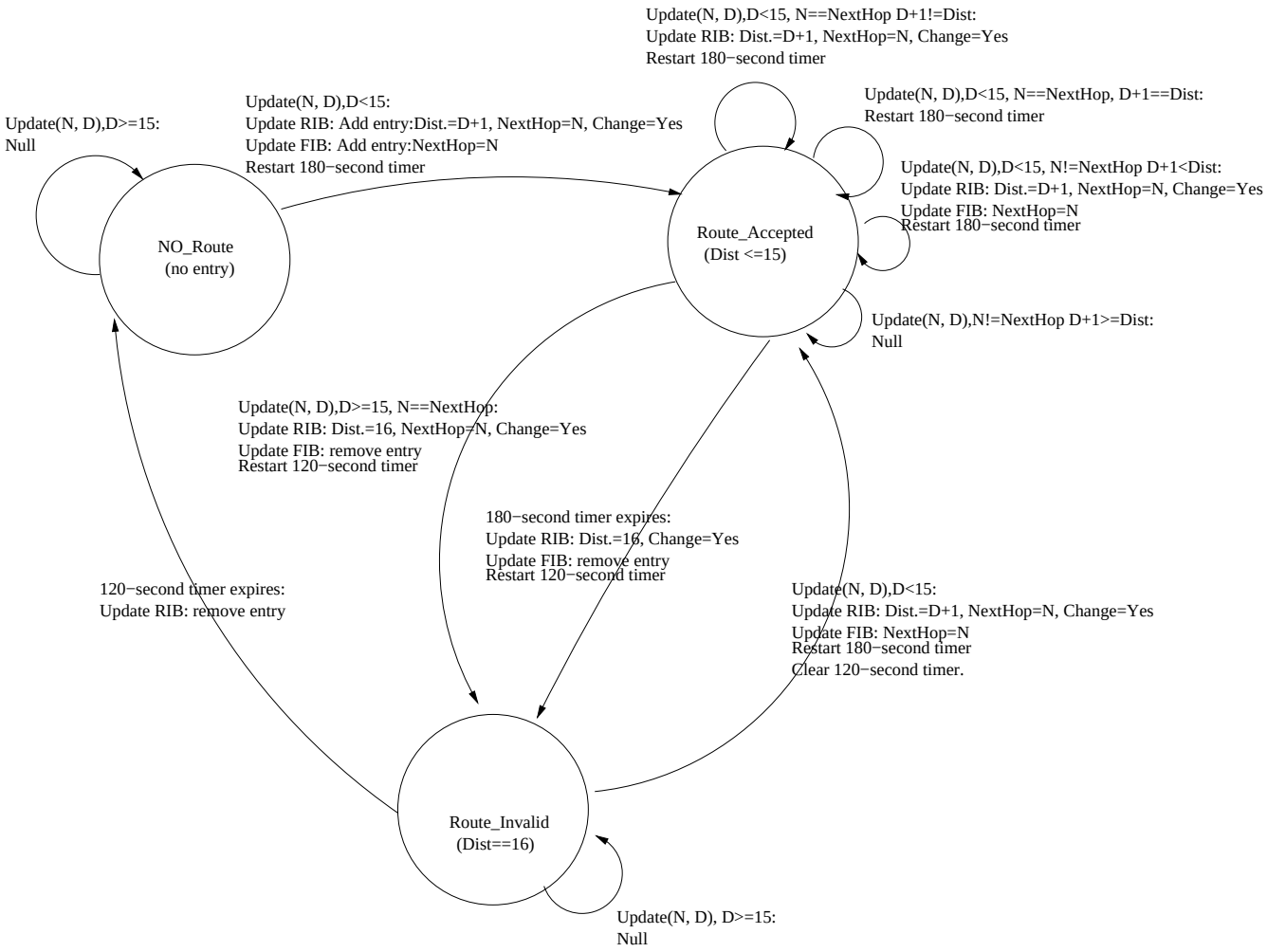

Fig. 1. RIP route State Machine

2) Change flags should not be cleared when sending update upon a request.
RIP standard doesn't explicitly specify this, leaving room for implementation bug.
We noticed that [3] showed that in some scenarios AODV standard version 2[8], allows loops to be formed. The authors attribute this failure to that fact that the standard fails to anticipate some sequence of events that consequently leads to the loop. We expect, if FSM state machines are developed for AODV, the ambiguities could also be identified.

APPENDIX

Appendix
proof of sufficiency

The goal of the RIP protocol is to compute a table at each router which provides the shortest path to destination d. We will show that if every router satisfies (R1), (R2), (R3), and (R4), it is sufficient for the convergence of RIP protocol: the routers will eventually obtain a shortest path to d, starting from an arbitrary state.

RIP is substantially different from Asynchronous Bellman-Ford Algorithm for several reasons:

- RIP allows the implementations to decide how to keep the update learned from neighbors
- RIP has a maximum metric(16) which is treated as infinity.
- A RIP router must apply split horizon.

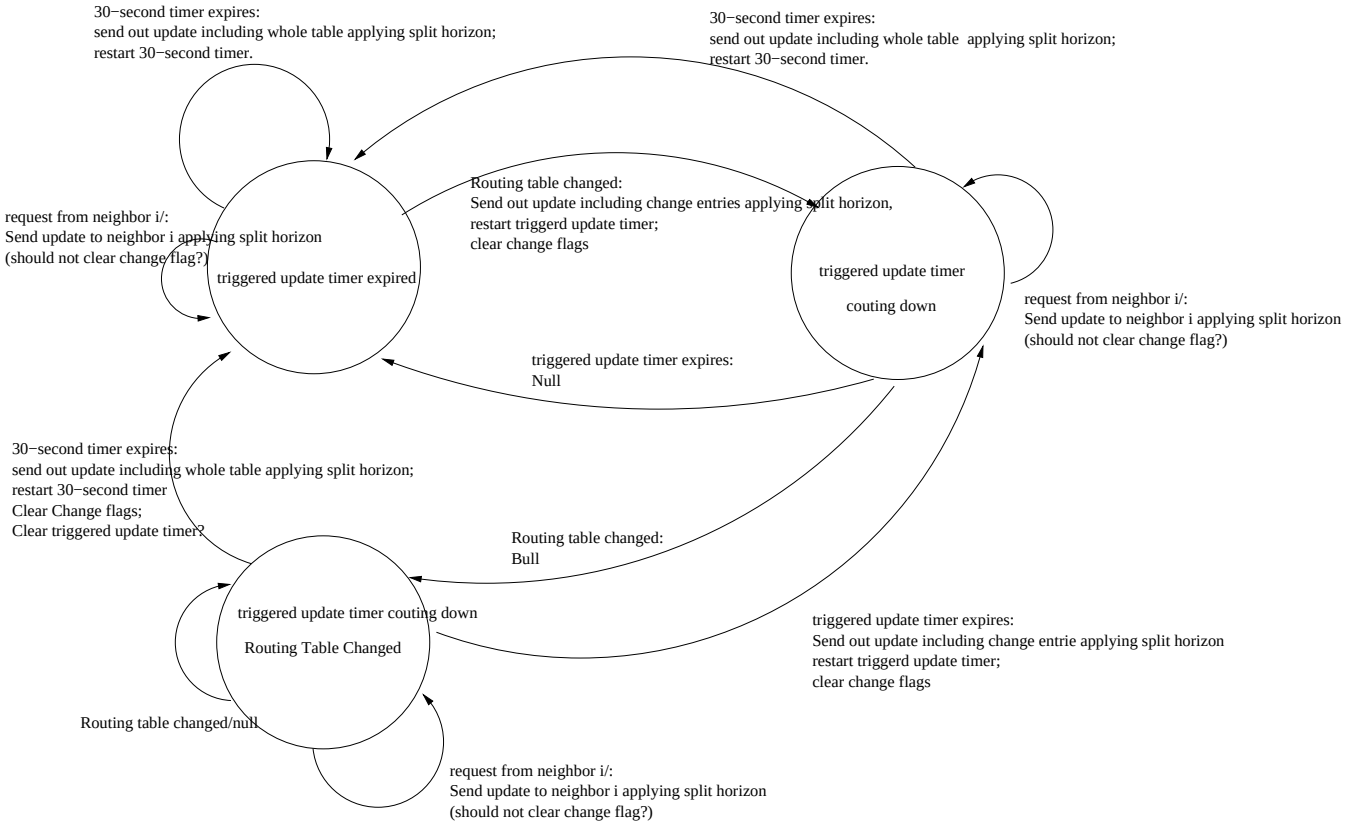

Fig. 2. State machine for the RIP output process

Assumption 1: A number of network topology changes occur up to some time T_0 and no other changes occur subsequently.

Assumption 2: Every router in the network satisfies (R1), (R2), (R3) and (R4).

Assumption 1 and Assumption 2 are assumed in the rest of this section.

Assumption 3: Immediately after an update is sent out from router a to b, b will receive it. That is, in the network there is no update that has been sent out by the sender router but has not received by the receiver router. Assumption 3 may be too strong but it will make our proof easier and (hopefully) we will remove this assumption later.

Definition 1: δ_r is the correct shortest distance from router r to the destination after time T_0.

According to the definition of $D_r(i,T)$, it is clear that: $\forall r, \delta_r \leq 16$.

Lemma 1: There is a time $T_g \geq T_0$ such that:

$\forall r, \forall T > T_g, Dist_r(T) \geq \delta_r$.

Proof:

$$\theta(T) = \begin{cases} 0, \text{if for all } r, Dist_r(T) >= \delta_r; \\ \min_{\forall r}\{Dist_r(T) | Dist_r(T) < \delta_r\}, otherwise. \end{cases} \tag{1}$$

$$S(\theta, T) = \{r | Dist_r(T) = \theta \ AND \ Dist_r(T) < \delta_r\} \tag{2}$$

$$Number_S(\theta, T) = \text{number of routers in } S(\theta, T). \tag{3}$$

If $S(\theta(T_0), T_0) = \phi$, then for all r, $Dist_r(T_0) \geq \delta_r$. According to (R4), all the distances a router advertises to its neighbors are longer than or equal to its real distance. As a result, after T_0,none of $Dist_r(T)$ will decrease to lower than δ_r. The lemma holds.

If there is such a router $s \in S(\theta(T_0), T_0)$, because $Dist_s(T_0) < \delta_s \leq 16$, so $Dist_s(T_0) \leq 15$. Then after T_0, if s receives a distance from a router $r \notin S(\theta(T_0), T)$, it will not accept the distance if $r \neq Nexthop_s(T)$

because $D_s(r,T) = min\{16, Dist_r(T) + C_s(r)\} \geq \delta_r > Dist_s(T)$; if $r = Nexthop_s(T)$, s will accept the distance and $D_s(r,T) \geq \delta_r > Dist_s(T_0) + 1$.

If s receives a distance from a router $u \in (S(\theta(T_0), T_0)$, this distance must be at at least $Dist_s(T_0)$ according the definition of $S(\theta, T)$.

According to (R1), $Dist_s(T_0) = D(Nexthop_s(T_0), T_0)$. According to (R4), after T_0, $Nexthop_s(T_0)$ will either send s an update which is longer than or equal to $Dist_s(T)$, or an infinity distance is sent to s because $Nexthop_s(T_0)$ uses s as the nexthop(no matter simple split horizon or split horizon with poison reverse is used), both within 180 seconds. In both cases, according to (R1),(R2),and (R3), $Dist_s(T) > Dist_s(T_0), or\ Dist_s(T) = 16$. That is, at time $T_1 = T_0 + 180$ seconds, $Dist_s(T) >= Dist_s(T_0) + 1$. Meanwhile, routers in $S(\theta(T_0), T)$ may be used as nexthop by some other routers, but their $Dist$ will be at least $\theta(T_0) + 1$, so these routers will never become a member of $S(\theta(T_0), T)$. Therefore, the number of routers in $S(\theta(T_0), T)$ will also decrease at least by one after T_1. Therefore, after $T_2 = Number_S(\theta(T_0), T_0) * 180$ seconds, $S(\theta(T_0), T) = \phi$, and $\theta(T_2) \geq \theta(T_0) + 1$.

This procedure repeats and since there are only finite number of routers in any networks, and $\forall r, Dist_r \leq 16$.) after finite time T_g, the network will reach the state when $\theta(T_g) = 0$. That is, $\forall r, Dist_r(T > T_g) \geq \delta_r$.

The lemma holds.

Theorem 1: If every router in the network satisfies (R1), (R2), (R3) and (R4), There is a time $T_{converge} \geq T_0$ such that: $\forall r, \forall T > T_converge, Dist_r(T) = \delta_r$.

Proof:

Induction on the k=hops that a router is away from the destination when using the shortest path. Induction Hypothesis at most $180k$ seconds after routers directly connected to the destination knows the shortest path, the routers $k+1$ hops away will know the shortest path, and will accept the route and won't change anymore.

$k = 1$, routers one hop away from the destination is directly connected to the destination. After T_0,each router directly connected to destination d will set the its corresponding $Dist_T$, so the Induction Hypotheses is true for k=1. Each such router should not use other routers as its $Nexthop_T$. According to (R4), it will send at least one update to each of its neighbors within 180 seconds.

Now suppose the Hypothesis is true for $k+1$. After the routers k+1 hops away know the shortest path, their nexthops should be one of the routers k hops away from the shortest path. Therefore, each of them will send at least one update to each of its neighbors excluding its current Nexthop within 180 seconds according to (R4). For each router r which is $k+2$ hops away from the destination, $D_r(< the(k+1) - hopneighbor >, T) = \delta_r$. According to Lemma 1, after T_g every router in the network satisfies: $Dist_r(T) \geq \delta_r$. So, according to ((R2), r will accept the route. Since the routers k+1 hops away won't change their route, according to (R3), it is impossible for a router k+2 hops away to increase its distance. It is also impossible for a router k+2 hops away to decrease its distance once it accept the correct value, since after T_g, for all r, $Dist_r(T) \geq \delta$. so the Induction Hypothesis is true for k+2.

The theorem holds.

Sketch of the proof of necessity

(R1), (R2), (R3) and (R4) are all required by RIP standard. We will show that any combination of the 3 are not sufficient to guarantee RIP convergence by giving counter example.

- A RIP router always chooses 1 as the best distance, with an arbitrary nexthop, , violating (R1). But (R2),(R3) and (R4) are satisfied. Since some router chooses to use it as the nexthop, the network will never converge to the correct state.
- A RIP router always chooses the route from one particular neighbor, even if this neighbor is never on the best route,violating (R2) but satisfying (R4). (R1) is satisfied since the route chosen is a valid one, while 3 is also satisfied since $Nexthop$ doesn't change at all.
- A RIP router always chooses the last route learned, possibly violating (R3). (R1 and (R2) are satisfied, in some scenarios the network won't converge.
- A RIP router is in the middle of the best path to the destination, but never advertises its routes. so the router that would use it as the nexthop will never knows the route, and would choose a non-shortest route instead.

REFERENCES

[1] D. Bertsekas and R. Gallager. *Data Network*. Prentice-Hall, 1992.

[2] K. Bhargavan, C. A. Gunter, and D. Obradovic. Rip in Spin/Hol. In *Theorem Provers for Higher-Order Logics*, Aug. 2000.

[3] K. Bhargavan, D. Obradovic, and C. A. Gunter. Formal Verification of Standards for Distance Vector Routing Protocols. In *Proceedings of ACM Sigcomm*, Aug. 1999.

[4] S. Bradner. Key Words for Use in Rfcs to Indicate Requirement Levels. RFC 2119, SRI Network Information Center, Mar. 1997.

[5] D. Brand and P. Zafiropulo. On communicating finite-state machines. *J. ACM*, 30(2):323–342, 1983.

[6] T. Griffin and G. Wilfong. A Analysis of BGP Convergence Properties. In *Proceedings of ACM Sigcomm*, Sept. 1999.

[7] G. Malkin. Routing Information Protocol Version 2. RFC 2453, SRI Network Information Center, Nov. 1998.

[8] C. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (aodv) routing. http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt, February 2003.

[9] Y. Rekhter and T. Li. Border Gateway Protocol 4. RFC 1771, SRI Network Information Center, July 1995.

[10] K. Varadhan, R. Govindan, and D. Estrin. Persistent Route Oscillations in Inter-Domain Routing. Technical Report 96631, SRI Network Information Center, Feb. 1996.