RIP: Protocol Overview and Xorp Design

Orion Hodson
International Computer Science Institute

August 2003

Introducing the Routing Information Protocol

- An Interior Gateway Protocol.
- Based on distance vector
  (Ford and Fulkerson, Bellman-Ford).
- Multiple variants on RIP (XNS, IPX, IP).
- IP variants: RIPv1 (deprecated), RIPv2, and RIPng.

Talk Outline

❋

- Distance Vector 101
- RIP versions, docs, deltas
- Proposed Xorp Design

Distance Vectors

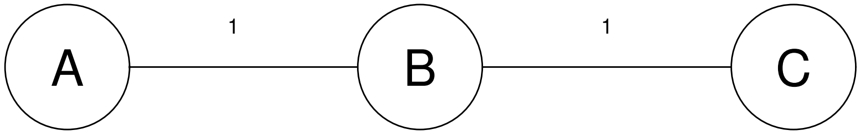
Nodes build and maintain a table of distances to other nodes and exchange this with their peers.

Information received from peers is used to update table of distances at receiving node. Receiving only receives data from immediate peers and knows their distance.

Updates are sent periodically.

Updates are SIMPLE ➜ peer (or network) plus distance.

Classic Problem: Counting to Infinity



Distance from C	A	B	C
❶ Stable point	2	1	0
❷ Link BC fails	2	??	
❸ A sends update (C at distance 2)	2	3	
❹ B sends update (C at distance 3)	4	3	
❺ A sends update (C at distance 4)	4	5	
❻ . . . ad infinitum . . .	. . .	. . .	

➜ Pick a small distance and call it infinity.

Hold-down

In case of path failure, advertise path as infinity,
and wait for "hold-down" interval.

Okay, iff information reaches all nodes before hold-down timer expires.

May slow convergence and does not solve count to infinity.

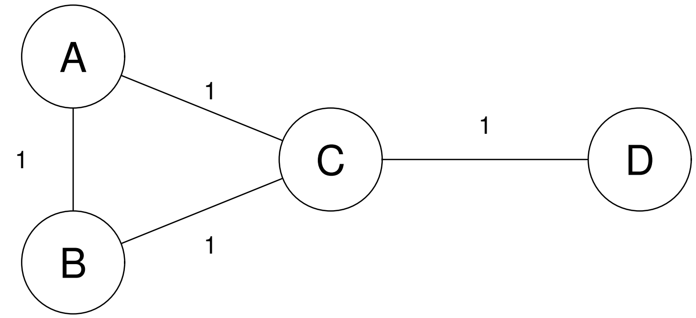
Split Horizons

Split Horizon	Split Horizon with Poison Reverse
Don't advertise information back to its source	Advertise information back to source with cost of infinity

Speeds up convergence in some cases (eg, A–B–C).

Solves some, but not all counting to infinity problems.

Counting to Infinity with a Split Horizon



Distance from D	A	B	C	D
❶ Stable point	2	2	1	0
❷ Link CD fails				
❸ C believes D unreachable (SH)	2	2	∞	
❹ C sends update to A and B				
❺ C's update reaches B	2	∞	∞	
❻ A sends periodic update	2	3	∞	
❼ C's update reaches A	∞	3	∞	
❽ B sends periodic update	∞	3	4	

Triggered Updates

Fast propagation of changes (particularly for deleted nodes/links).

- Speeds convergence.

RIP as a Distance Vector Protocol

- Infinity fixed at 16.
- Hold-down mandatory.
- Split Horizon and Poison Reverse are recommended options.
- Triggered updates mandatory for deleted routes, optional for new or changed routes.

RIP: Some practicalities

- Timers to age and timeout routes.
- 2 Packet Types: Request and Response.
- Timer randomization.
- Optional use of authentication.

RIP Default Timer Values

Timer	Period (s)
Route Expiry	180
Route Garbage Collection (hold-down)	120
Periodic Updates	25–35
Triggered Update	1–5

RIP Request Messages

Two variants:

Whole Table
Used at start-up and response employs split horizon processing

Specific Routes
Used for debugging and response does not employ split horizon

RIP Response Messages

Query Response
Response to a specific query

Regular update
Contains all routes

Triggered update
Contains route updates since last update (regular or triggered)

Triggered updates are blocked by regular updates.

Multi-Packet RIP Response Messages

❋

Packets hold finite number of route entries (25 for RIP on IPv4). A Response message will typically be composed of multiple packets.

Most vendors send trains of response messages with some small inter-packet spacing to avoid buffer overflow. 10–50ms is typical.

RIP docs

RFC	Contents
2453	RIP version 2
2080	RIPng for IPv6
1721	RIP version 2: Protocol Analysis
1722	RIP version 2: Applicability statement
1723	RIP version 2: Carrying additional Info
1724	RIP version 2: MIB Extension
2082	RIP version 2: MD5 Authentication

RIP Protocols

RIP v1: Class based, designed with expansion in mind. UDP transport. Fixed maximum packet size.

RIP v2: Extension of RIP v1. Classless. Supports tagged routes, next hops, authentication. Implementations typically interoperate with RIPv1.

RIPng: As RIPv2, but relies on IPv6 for authentication mechanisms and v1 equivalent interop. Must have link-local addresses in packets as source addresses and nexthops must have link-local addresses. May use path MTU discovery for packet sizes.

First Cut Features

Functional RIPv2 and RIPng implementation
Highly similar ➜ templates with limited specialization

Scale to O(10000) ➜ small state including a timer per route

Support tunable timer abd packet spacing values

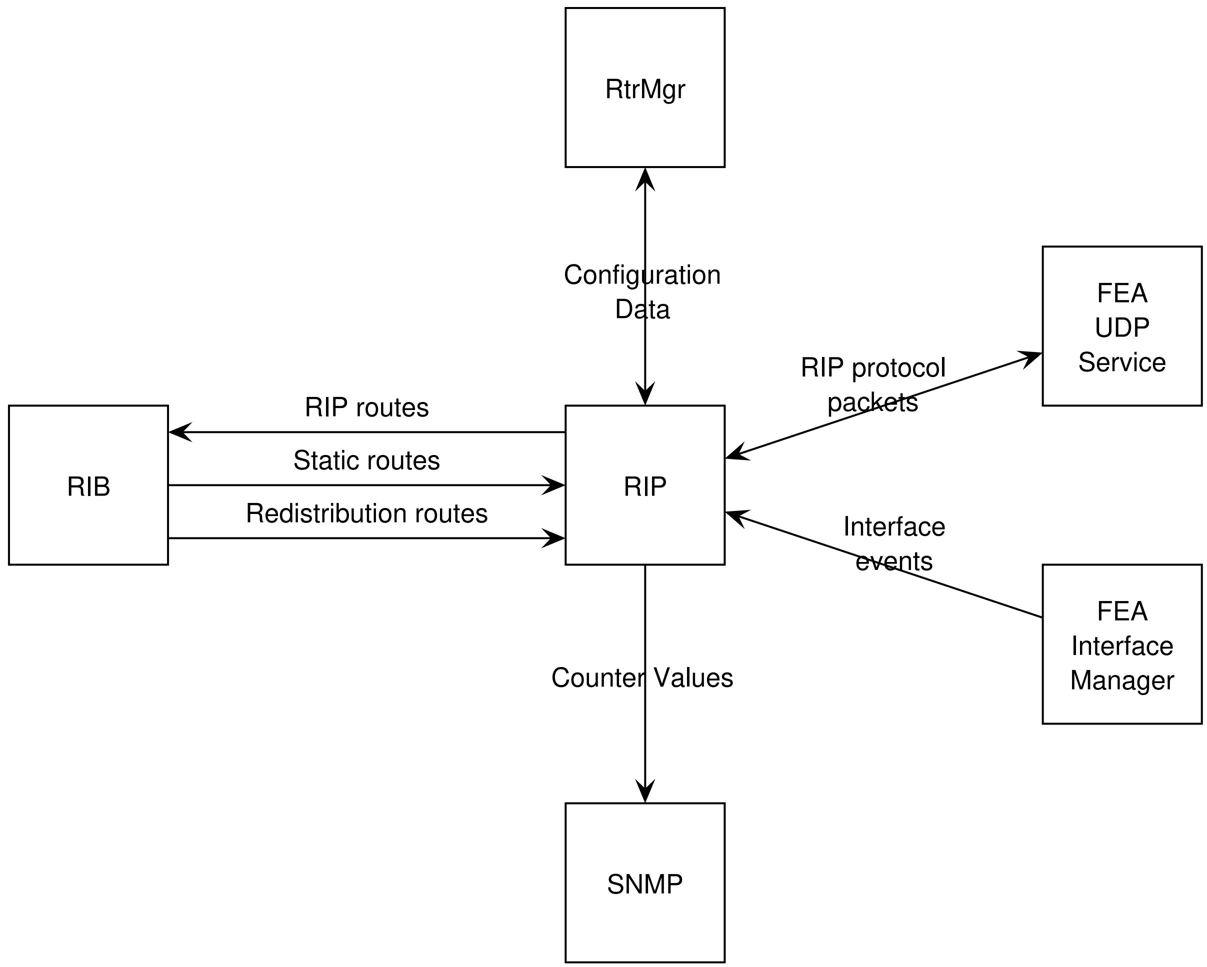
SNMP support

Later Work

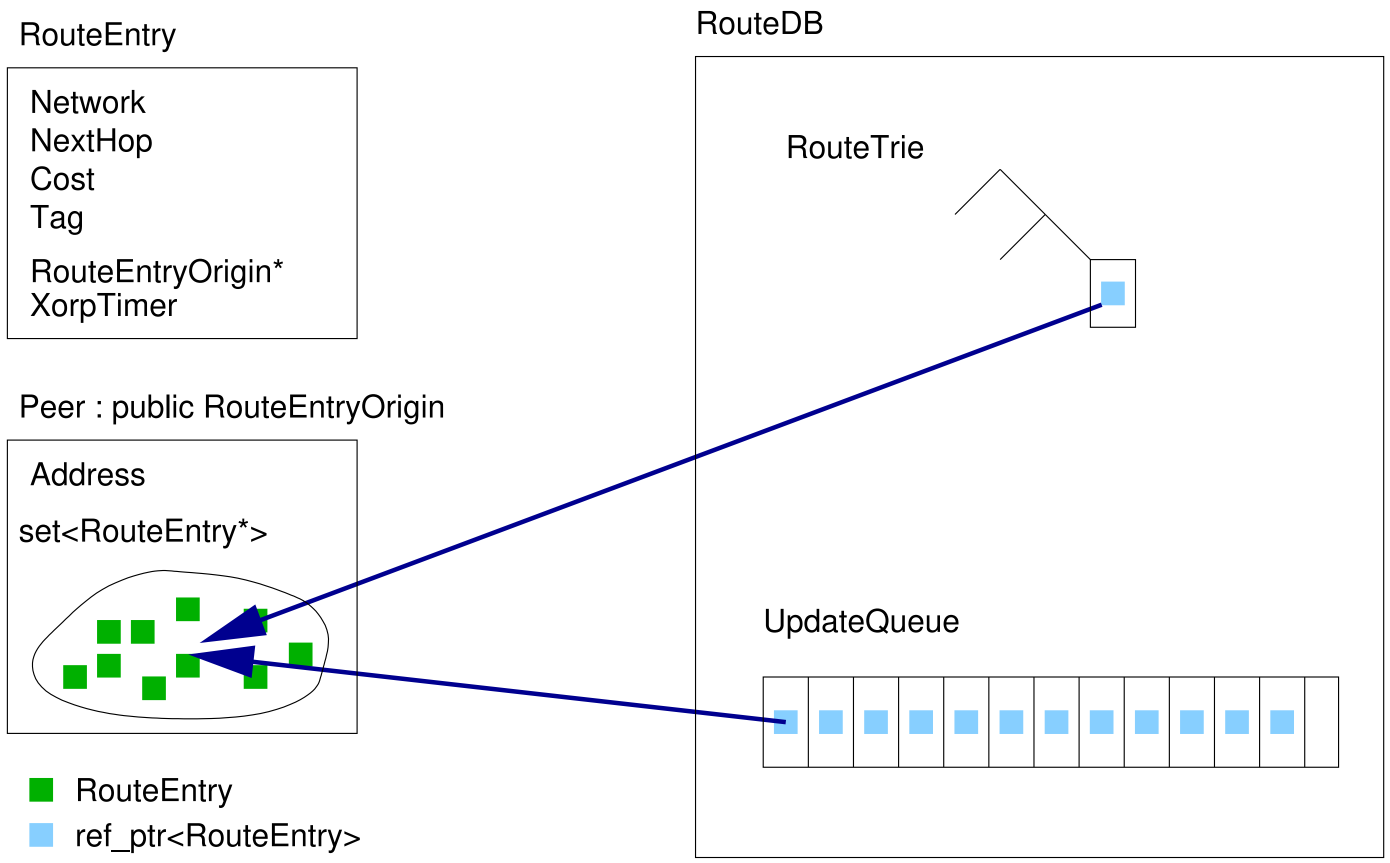
Tagged route filtering to help manage route redistribution.

RIPv1 and RIPv1 Inter-Op.

RIP Interaction with Xorp processes



At the core: Peers, Routes, RouteDatabase



Route Management

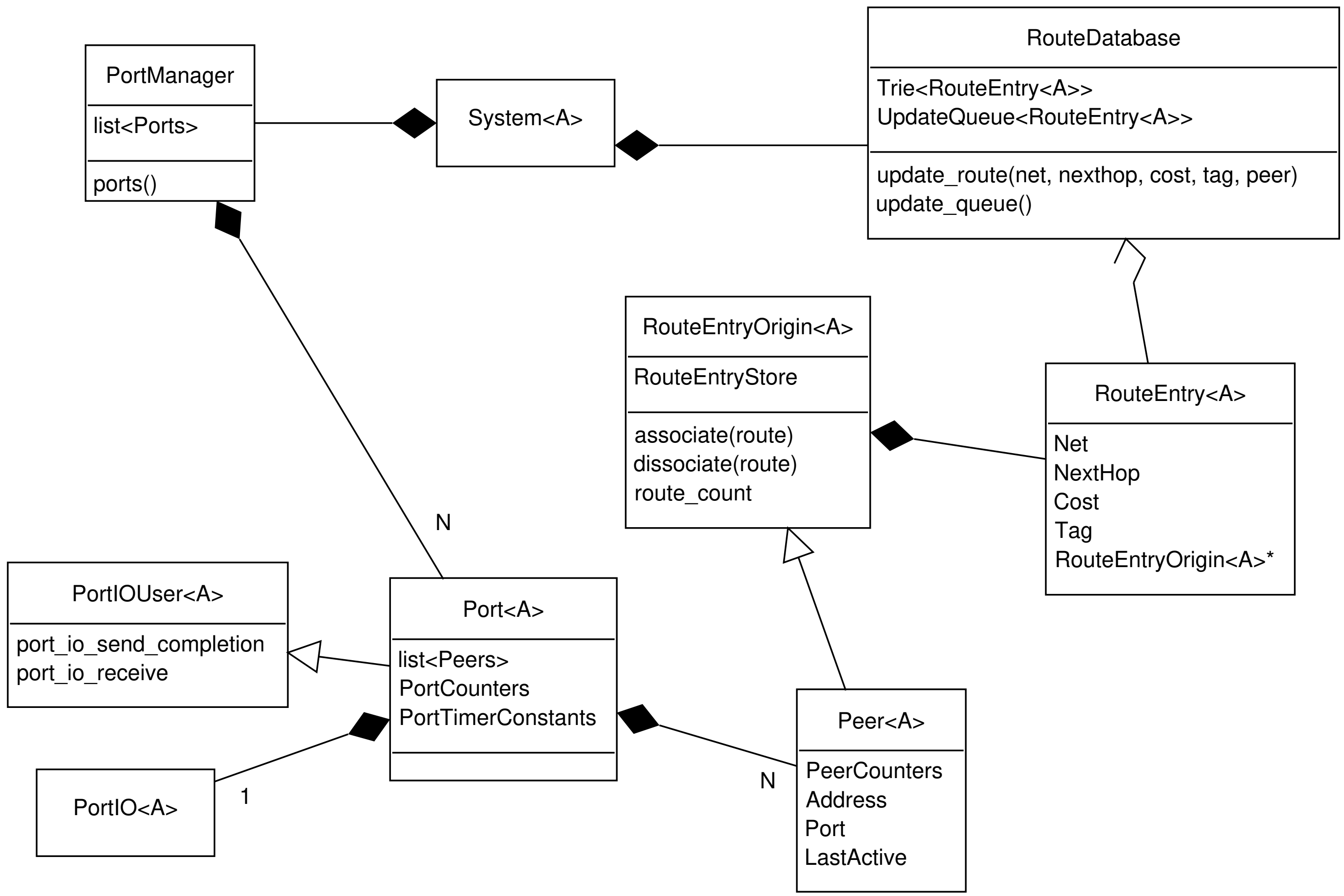
`RouteEntryOrigin` objects own `RouteEntry` objects. `RouteEntry` objects associate and dissociate themselves from `RouteEntryOrigin` on construction and destruction.

`RouteDB` is shared `RouteEntry` store - contains `RouteDB::RouteTrie` and `UpdateQueue`.

`RouteDB::RouteTrie` is used for lookup and modify operations. `RouteEntryOrigin` objects may be used for table "dump" operations (much faster).

`UpdateQueue` is used for triggered updates and is a vector reference counted `RouteEntry` objects. Route can be removed from Trie, and just exist in the may exist in `UpdateQueue`.

Core Classes and their Relationships



Port Objects

A `Port` potentially exists for each Xorp VIF and bound to an address on a VIF.

`Port` objects manage 0...N `Peer` objects.

`Port` objects are instantiated by `PortFactory` instances and managed by the `PortManager` object.

Port Objects: Input and Output Processing

❊

- Receives request and response messages from `PortIO<A>`.
- Performs authentication.
- Feeds routes and updates into `Peer object` and `RouteDB objects` (with optional Split Horizon/Poison Reverse).
- Holds a read iterator to `UpdateQueue` and has a timer for triggered updates (walk read iterator to end of `UpdateQueue`).

Periodic updates

Periodic updates involve sending the entire contents of the RIP route database. Typically every 30 +/- 5s.

Two options:

- Perform periodic updates by trawling routes and handling all `Port` instances simultaneously. (Less work, correlated output).
- Perform periodic updates on a per `Port` instance basis. (More work, decorrelated output). [Preferred]

Per Peer Periodic Update [Proposed]

Each peer has a `PeriodicUpdater` class that iterates through list of peers and their sets of routes. It's timer driven and outputs 1 response packet per each time it's scheduled. The timer expiry interval is set to the interpacket spacing.

The `PeriodicUpdater` maintains a reference to the last route it puts in each response packet so iteration through a `Peer` objects set of `RouteEntry` objects can always resume from a valid point.

RIB Interaction

Input

`RIBPeer` class for storing routes learned from RIB. No Timers on these routes in `RouteDB`.

Output

`RIBOutput` class that is attached as a read-iterator to `UpdateQueue`.

DebugOutput [option]

`AnyTargetOutput` as RIBOutput, but more generic.

XRL Interfaces

Separate XrlTargets for IPv4 and IPv6 RIP systems.

Interface details to be decided.

All objects reachable from Top-Level `System` object.

SNMP counters are (mostly) in place.