

# CS168 VAN – Virtually Active Network IP and Routing

---

*Assignment Out:* September 25, 2007  
*Design Meeting Due:* October 1, 2007  
*Assignment Due:* October 10, 2007 (11:59 pm)

---

## 1 Introduction

For the first part of the VAN project, you are to take a simple network and introduce dynamic routing. This will include detection and interpretation of the network topology on startup, the ability for nodes to send and receive specific types of messages, dynamic reconfiguration of route tables and a system like IP forwarding. Lectures 1-4, chapter 5 from Tanenbaum, will be especially helpful with this part of the project. In addition, you should read the online guide to the VAN system (<http://www.cs.brown.edu/courses/cs168/van.html>) found on the course web page. In fact, you should read it before you read this, since things will be clearer. Go read that, and then come back when you're finished.

As it says in the VAN guide, you need only a few files to get set up. The first thing you should do is make your own `netconfig` file (this could just be a copy of the default version). You'll want to design your own (somewhat complex) network so that you can test various cases, and exercise your code to its fullest. You also need to choose what language you want to use. Both C and C++ will work, so it's really a matter of preference. In `/course/cs168/asgn/ip` there is a sample Makefile to build your library code and your driver program. You don't need to use it, but if you want something to get you started, or don't want to create your own Makefile from scratch, feel free to copy these over as starting points. In that directory, there is also the beginnings of a driver which you can use as a starting point. Finally, if you're doing this project in C++ you will need to copy the `vancpp.C` and `vancpp.H` files from that directory into your project directory.

As this is a 2-person group project, you should find a partner to work with. Once you have, email `cs168tas@cs.brown.edu` to inform us of your pairing. If you are having problems with this (there might be an odd number of people in the class this year), post to `cs168@list.cs.brown.edu` or email `cs168tas@cs.brown.edu`. Once the groups are set, you'll be assigned a mentor TA to help you through both IP and the next project, TCP.

## 2 Requirements

Before you start coding, you need to understand what you're doing. There are a few important pieces to this assignment. The first involves designing routing tables and headers. Routing tables

are used when forwarding data and when furnishing adjacent nodes with route information. Headers are used by your code to identify certain messages, src/dest pairs,<sup>1</sup> etc. Next, the nodes need to be able to provide the send/recv interface to the next layer up. This interface also needs to use the route-table information to implement packet forwarding. Either packets are queued locally, or they are passed on to the next node in the path. Finally, a RIP-like system is added so that the nodes do dynamic network detection. This involves passing messages and pinging adjacent nodes.

## 3 Implementation

### 3.1 Network Layer

The first thing you write is the network layer. This contains a `van_node` (C) or a `VanNode` (C++), and provides the same send/recv calls that the van node exports, giving higher layers access to the network through your code (in this assignment, “higher layers” will just be some driver program, but the next 2 parts of the project will build actual layers). Also contained in this layer are your route tables. They need to be flexible, since the network can change. They are used by each node to understand the topology around them. They contain information such as interface-to-node mapping, shortest paths, etc. Since you’ll be doing a lot of lookups in these tables, make sure to make them fast!

Next, you define packet headers. The nodes have to be able to send and receive special messages, and these are defined by global packet headers. Think carefully about these. They’ll be passed around constantly in your network, so making them as small as possible is crucial. Take a look at some real world headers (particularly those for IP), and try to think ahead to what you will need for TCP.

Just as the VAN node queues incoming data, your code should do the same thing (since the upper layers won’t always be reading data as it arrives). You should set up a “listening thread” to queue up incoming data. The support code provides a thread-safe, blocking queue. Take a look at `/course/cs168/include/bqueue.h` for more information. Using information provided in your headers, your code determines whether each packet was meant for the current node, or another one. If the destination is the current node it should be queued so the next layer up can read the packet, otherwise forward it on to the next node in the path.

Before working on the core part of this assignment, you should test your supporting code. Try creating a static network (hard code it, read from a route table, etc.) and make sure that your code works. Send data from one node to another one that requires some amount of forwarding. This isn’t a requirement, but it’s not a bad idea to sanity check your system like this.

### 3.2 RIP

Now write the processing code that makes the network dynamic. Start with the RIP protocol described in lecture to add route discovery to your system, so that when a node is created, it starts

---

<sup>1</sup>You can simply use the VAN node number for addressing.

building its tables from the surrounding nodes.<sup>2</sup> (note: you are free to try something like OSPF instead and get extra credit, but this is harder). Using RIP includes implementing the loss of route protocol to detect sudden problems and route around them (think: periodic pinging). You'll also need a timeout value associated with data, so that if you can't reach a node within a certain amount of time, you just give up. To make your RIP complete, you are required to implement split horizon, poison reverse, and triggered updates. Again, refer to the lecture notes and the texts for specifics on how these protocols work. For a more verbose description, check out the RIPv2 RFC (2453) found in the links section of the homepage.

### 3.3 Driver

Finally, write a driver program (actually, writing this at the start will give you a good way to test as you code). This has to demonstrate all the features of the system. For example, you should be able to instruct a node to send or receive, tell an interface to go up/down, etc. Printing out routing information at each node will show the path that data is taking, what kind of messages are being passed and will let you test things like dynamic route changes (and will let you demonstrate to us that your code really is doing what it should be doing). Anything else that you think is important, or will help test/demo a feature of your system should get added to your driver.

## 4 Getting Help

This project isn't supposed to be too painful, and you have many resources to help you. Make sure you've read this handout and the VAN introduction. This handout, as well as the lecture notes, will be available on the web page. The mailing list ([cs168@list.cs.brown.edu](mailto:cs168@list.cs.brown.edu)) is always a good place to get help on general topics, and the TAs will, of course, be holding TA hours.

Since this is a group project, you'll have your group member to bounce ideas off of. Make sure that you work together, and try to split the project up so that neither of you has too much to handle<sup>3</sup>. You are allowed to share any and all code with your partner. You are not, however, allowed to share code with other groups. There is a non-collaboration policy for this class, but you've all been in some classes, and you know the drill. You can talk to other groups about concepts, algorithms, etc., but the code has to be written by each group on their own.

Finally, each group will have a mentor TA. This means that you'll have one of the two oh-so-wonderful TAs as your group's advisor. You'll need to set up an appointment to meet with your mentor TA for some time this week to discuss the project design. Once you've got the ok on this, you should stay in contact with your mentor, who will be grading your project and will be able to explain what the project ultimately should be doing. Your mentor also will do his best to help outside of TA hours, debugging, discussing design, etc. Just because your mentor is helping you out, however, doesn't mean that he/she is at your beck and call. Understand that the TA staff

---

<sup>2</sup>Use the ping time of the link as your metric.

<sup>3</sup>An easy way to split things up is for one person to implement routing (RIP/OSPF) and the other to be responsible for everything else (packet forwarding, send/rcv interface, etc), but of course you can do whatever you feel is appropriate

is busy too, and while they'll try to help you as much as possible, there may be times when they simply won't be able to.

## 5 Extra Credit

There are loads of areas for potential extra credit options for this assignment. Probably the easiest thing you can do is to implement a different routing algorithm (such as OSPF). Various ICMP extensions are also possible and pretty easy, as well as multicasting (for which you'll also need to implement IGMP) if you're feeling particularly masochistic.

It should be noted that **you are only required to use p2p links** for this assignment. Use of **eth** links substantially complicates things but gives you many more options for extra credit. You'll need to implement ARP (and maybe proxy ARP), but then you can do routing with subnetting, some sort of dynamic configuration protocol if you want, similar to dhcp/bootp (or you can punt on this with a static configuration file, but you need to somehow set up a default router at the very least). If you plan on giving it a try, we suggest that you design for this carefully and first get p2p links working.

## 6 Handing In and Grading

Once you have completed the project you should run the electronic handin script `/course/cs168/bin/cs168_handin ip` to deliver us a copy of your code. This should be done by the project deadline or in the next couple of days if you want to use your late days. Your mentor TA will arrange to meet with you for your interactive grading session to demonstrate the functionality of your program and grade the majority of it. This meeting will take place at some point shortly after the project deadline. Between the time you've handed in and the demo meeting, you can continue to make minor tweaks and bug fixes. However, the version you've handed in should be nearly complete since it could be referenced for portions of the grading.

## 7 Conclusion

You should start on this project now. We expect all of the projects in CS168 to take the full amount of time we give you. It can be tricky so we want to make sure that you stay on top of it. You have to have your first design meeting with your mentor TA by Oct 1st. For this meeting you should have a clear sense of what your program is going to look like and a complete list of what you don't understand. So... start talking with your partner right away, and get ready to get connected!