Virtual Circuits

A *virtual circuit* is one which appears to the user as equivalent to a dedicated point-point service but is maintained by computers. It will usually transport data at a guaranteed rate (bit/s) and with guaranteed reliability and error rate.

Internally information is carried by many small packets, each with a short *virtual circuit number* which identifies its virtual circuit over that physical link. The virtual circuit changes as the packet is switched by a switch or router from one incoming link to an outgoing link, according to information held in *routing tables*. It is the combination of entries in the routing tables of successive switches which defines the end-to-end virtual circuits.

Before discussing routing there are several terms to be discussed.

1. A *switch* and *router* are for our purposes very similar devices. Each router normally connects to several other routers and can divert messages coming from one link to any other link.
2. A *link* is a long distance connection between two routers, usually at least several kilometres long (perhaps many thousands of kilometres). It provides a point-to-point connection and may carry thousands of user circuits.
3. A *circuit* is what the user sees as a connection to another user. The circuit is first *established* by a *call establishment* request which contains the full end-address of the called user (IP or similar, equivalent to a telephone number). This sets up a suitable set of entries in the switch *VC routing tables*. After establishing the circuit, the user may just send data over it according to any suitable protocol. Finally a *call disconnect* will break or tear down the connection by clearing its routing table entries.
   Alternatively, if you are emphasising the hardware level, a circuit may be equivalent to a link as described above.
4. A *port* is a physical connection by which a link connects into a switch or router. It usually means a connector for the cable (the physical aspect of the link) and associated interfaces to encode and decode data and transfer it to and from the router memory and processing.
   This is different from a **port** between protocol layers, such as from IP into TCP and other services.
5. A virtual circuit number (VC, or Virtual Circuit ID) is the number within the packet which identifies the virtual circuit. It usually changes as the packet goes from router to router over the network. Virtual Circuit numbers may be shared between different links to the same router, and even in different directions over the same link, but must be unique in one direction over the one link.

6. Virtual circuit number generation.

Virtual circuit numbers are usually quite arbitrary (except that some very small ones are often reserved for system work and communication between adjacent routers). Many systems have rules for allocating numbers.

In the example here, the station or router making or forwarding the call request generates both VC numbers, using a small value for the “outward” circuit and large ones for the “inward” circuit. Sometimes the router receiving the request might generate both, or it might receive the outgoing number and respond with the incoming number. What must be avoided is the situation where a left-to-right call request and a right-to-left call request select the same number for say left-to right signalling on the same link. This is handled by using the different blocks of numbers.

1. Routing tables

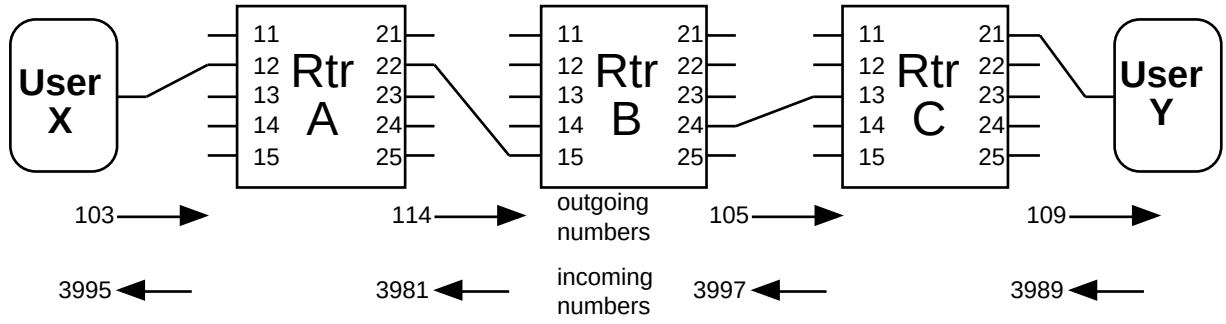
There are two types of routing tables.

1. **End-to-end routing** tables are used during call establishment (and are identical to the routing tables used all the time for datagrams). They are interrogated by the final address of the called user and give the best route toward that user, usually just specifying the output port to be used.

2. **Virtual Circuit routing** tables are set up from the route discovered by call establishment. For each packet, they map from {input_port, input_VC} to {output_port, output_VC}.

3. Setting up a virtual circuit.

4. For this assume that User X wishes to connect User Y.
   X has a connection to router A (via port 12, but this is irrelevant to X).



1. **User X** sends a call request to User Y, nominating 103 as the outgoing VC and 3995 as the incoming VC for the circuit. These values are random, as long as they do not duplicate numbers already used over the link. The values may be chosen by the station making the request (as here) by the station receiving the request (here Rtr A) or any agreed combination.
2. **Router A** finds that the best route to User Y is through port 22, forwards the call request over that port and sets up its VC routing table as below. Any packet with VC=103 on port 12 is given the VC=114 and sent over port 22. Similarly anything received with VC=3981 on port 22 is sent out on port 12 with VC=3995.
3. **Router B** receives the call setup request with VC=114 on port 15 (because this is its link to router A) and finds from the end-routing tables that the connection should be over port 24, to which it allocates VC=105. The randomly chosen VC number for the reverse circuit is 3997.
4. **Router C** receives the request with VC=105 on port 13 and finds that it should connect to port 21, for which it chooses a circuit of 109, with 2989 in the reverse direction.
5. **User Y** receives the request and accepts it, noting that for this circuit it will receive with VC=109 and send with VC=3989.

6. The final routing tables are shown in the table. These are what are needed for message transmission, but the table also shows a “reverse VC” entry. This is hardly ever shown in texts but is needed if the router must reply to a message. For example if Router B wants to reply to VC=114 on port=15, it will reply with VC=3997 (also of course on port 15).

	In_Port	In_VC	Out_Port	Out_VC	ReverseVC
Router A	12	103	22	114	3981
	22	3981	12	3995	103
Router B	15	114	24	105	3997
	24	3997	15	3981	114
Router C	13	105	21	109	3989
	21	3989	13	3997	105