

\*\*\*\*\*

ATM Forum Document Number: 97-0835

\*\*\*\*\*

**Title:** Proposed Appendix B of Performance Testing Baseline Text on Scalable Configurations

\*\*\*\*\*

**Abstract:** A standardized methodology for implementing scalable configurations for throughput and latency tests is presented.

\*\*\*\*\*

**Source:**

Arjan Duresi, Raj Jain, Justin Dolske, Gojko Babic  
The Ohio State University  
Department of CIS Columbus, OH 43210-1277  
Phone: 614-292-3989, Fax: 614-292-2911, Email: Jain@ACM.Org  
The presentation of this contribution at the ATM Forum is sponsored by NASA.

\*\*\*\*\*

**Date:** September 1997

\*\*\*\*\*

**Distribution:** ATM Forum Technical Working Group Members (AF-TEST, AF-TM)

\*\*\*\*\*

**Notice:**

This contribution has been prepared to assist the ATM Forum. It is offered to the Forum as a basis for discussion and is not a binding proposal on the part of any of the contributing organizations. The statements are subject to change in form and content after further study. Specifically, the contributors reserve the right to add to, amend or modify the statements contained herein.

\*\*\*\*\*

## Appendix B

### Methodology for Implementing Connection Configurations

#### B.1. Introduction

In Sections 3.1.7 and 3.2.7 of the baseline text, a number of configurations have been presented for throughput and latency measurements. In their basic form, these configurations require traffic generators and analyzers, whose number increases as the number of ports on a switch increases. Since the test monitors are rather expensive, it is desirable to define scalable configurations that can be used with a limited number of generators. However, one problem with scalable configurations is that there are many ways to set up the connections and the results could vary with the setup. In this appendix, a standard method for generating these configurations is defined. Thus, anyone can design a connection configuration for switches with any number of ports. Since the methodology presented here applies to any number of traffic generators, it can be used for non-scalable (basic) configurations as well.

Performance testing requires two kinds of virtual channel connections (VCCs): foreground VCCs (traffic that is measured) and background VCCs (traffic that simply interferes with the foreground traffic). The methodology for generating configurations of both types of VCCs is covered in this appendix.

The VCCs are formed by setting up connections between ports of the switch. The connection order of these ports is referred to here as a **VCC Chain**. For example, the VCC shown in Figure B.1 consists of one VCC chain passing through ports P1-P2-P3-P4-1. Another possible configuration for this "N-to-N" single generator scalable configuration would be P1-P3-P2-P4-P1. For an N-port switch, there are a total of  $(N-1)!$  possible configurations.

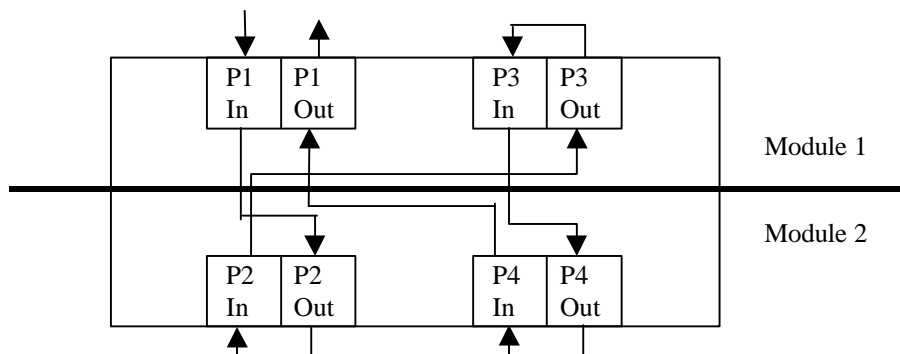


Figure B.1 One out of six possible VCC chains that can implement the 4-to-4 straight configuration with a single generator.

## Proposed Appendix B of Performance Testing Baseline Text on Scalable Configurations

If the four-port switch shown in Figure B.1 consists of two modules with two ports each, the measured performance may depend upon the number of times the VCC chain passes from one module to the other and may be different for different configurations.

At the end of this appendix, the pseudocode for a computer program is presented that allows generating a standardized port order for all connection configurations. This methodology (pseudocode) generally creates VCC chains that cross the modules as often as possible while still keeping the whole process simple.

### B. 2. Definitions and Rules

In order to generate a standard configuration, it is first necessary to have a standard method of numbering the ports of a switch. This method is presented in this section.

Consider a switch with several modules. Each module may have a varying number of ports. In order to number these ports, the first step is to generate a schematic of modules placed one below the other. The schematic is drawn such that the modules are arranged in a decreasing order of number of ports. Then the switch ports are numbered sequentially, along the columns, starting from the top left corner of the schematic. This port numbering helps in creating VCC chains that cross modules as often as possible. The port numbers obtained this way are represented by  $P_i$  in this appendix.

Figure B.2 shows one example of port numbering. The switch consists of three modules with 8, 7, and 6 ports respectively. The first port on the first module is numbered 1 or  $P_1$ . The first port on the second module is numbered  $P_2$ , and so on up to  $P_{18}$  as shown in the figure. For simplicity, we also refer to  $P_i$  as port  $i$ .

Module 1	[1] P1	[2] P4	[3] P7	[4] P10	[5] P13	[6] P16	[7] P19	[8] P21
Module 2	[1] P2	[2] P5	[3] P8	[4] P11	[5] P14	[6] P17	[7] P20	
Module 3	[1] P3	[2] P6	[3] P9	[4] P12	[5] P15	[6] P18		

Figure B.2 Numbering of ports in a switch with different number of ports per module.

The second thing we need is a standard method of presenting connection configurations. Each VCC chain is represented by a three-dimensional matrix  $C(i, j, k)$ . Matrix index  $i$  represents the interconnection order among the switch ports, where the value 0 indicates the source port and the last value indicates the destination port. Index  $k$  represents the generator number and index  $j$  represents the chain number starting at that generator. One

## Proposed Appendix B of Performance Testing Baseline Text on Scalable Configurations

row  $C(*, j, k)$  of the matrix represents a single VCC chain. For example, if the first VCC chain from generator #2 starts at source port P1, passes through ports P3, P4, P5, P6, P7, P8, and exits at port P2, the matrix C has the following entries:  $C(0,1,2)=P1$ ,  $C(1, 1, 2)=P3$ ,  $C(2, 1, 2)=P4$ ,  $C(3, 1, 2)=P5$ ,  $C(4, 1, 2)=P6$ ,  $C(5, 1, 2)=P7$ ,  $C(6, 1, 2)=P8$ ,  $C(7, 1, 2)=P2$ . Figure B.3 illustrates this VCC chain. The source port and destination ports are also represented by symbols  $C_{in}$  and  $C_{out}$ . For the VCC of Figure B.3,  $C_{in}=P1$ ,  $C_{out}=P2$ .

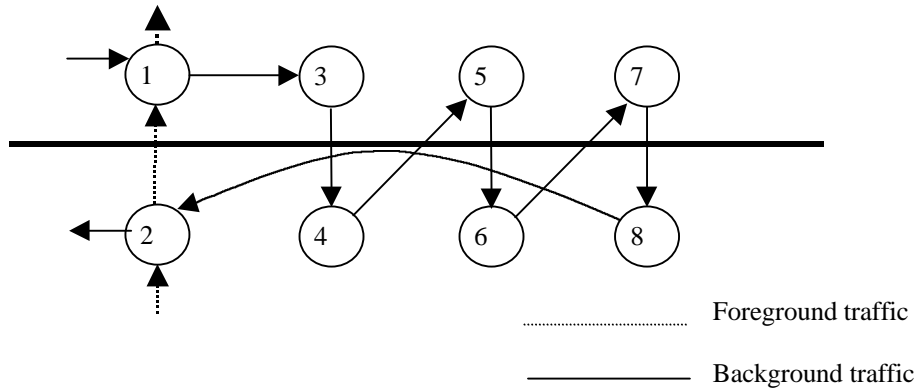


Figure B.3 A VCC chain

$NP(k)$  denotes the total number of **intermediate** ports for a VCC chain generated by generator  $k$ . Notice that the source and destination ports are not counted. In the case of Figure B.3,  $NP(2) = 6$ . Note that  $C(NP+1, j, k)$  is always the destination port.

For latency measurements, the foreground traffic involves only two ports, one for input and the other for output. To design the VCC chain for this traffic, the operator may simply chose any two ports, referred to as  $C_{Fin}$  and  $C_{Fout}$  respectively. Here, F in the subscript signifies "foreground." In order to avoid interference with the foreground traffic, the background VCC chains may or may not use  $C_{Fin}$  and  $C_{Fout}$ . If the background traffic does use these ports then it should only be in the directions opposite to that used by the foreground traffic. Figure B.4 shows a schematic representation of connection configuration for latency measurement of a 8-port 2-module switch. The foreground traffic uses ports P2 and P1 as the source and destination ports, respectively. So,  $C_{Fin}=P2$  and  $C_{Fout}=P1$ . The background traffic also uses these ports but in the opposite direction. Therefore, for the background traffic:  $C_{in} = C(0,1,2)=P1$  and  $C_{out}=C(0,1,2)=P2$ . The background traffic can use the six remaining ports in both directions. Incidentally, Figure B.3 shown earlier shows the VCC chain representation of this same configuration. From now on, we only show VCC chain representations for all configurations. It is straight forward to generate the schematic representations from it.

## Proposed Appendix B of Performance Testing Baseline Text on Scalable Configurations

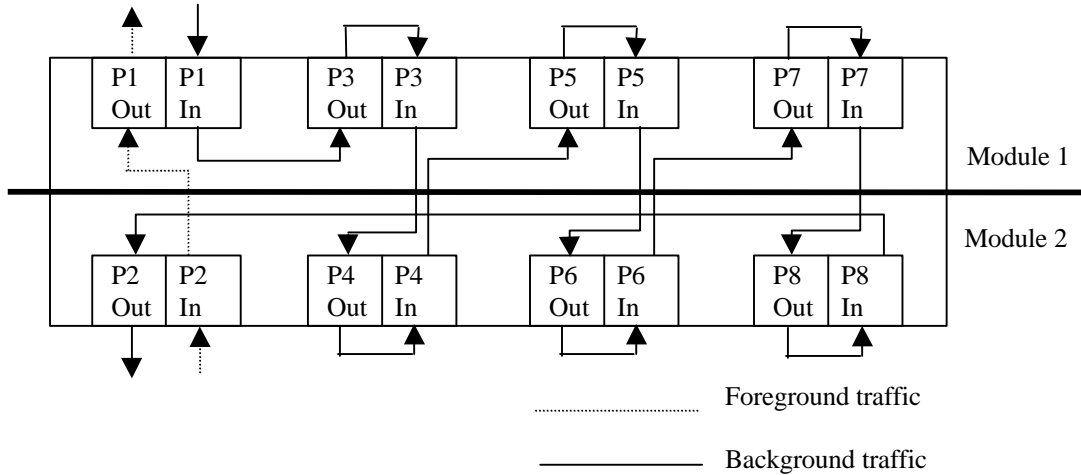


Figure B.4. A 7-to-7 straight configuration with one-generator for the background traffic.

### B.2. Connection Configurations Characteristics.

In this section we analyze several of the configurations for throughput and latency measurements and show how scalable version of them can be obtained using the algorithm given in Section B.3. The algorithm consists of three simple rules:

1. The chains generally go from port  $i$  to port  $i+1$  unless the port has already been fully used by other chains.
2. After generating  $j$ th chain,  $j+1$  chain can be generated simply by adding 1 to each port index of the  $j$ th chain.
3. If there are multiple generators, each generator uses a contiguous subset of the switch ports as source ports. Each generator needs as many source ports as the number of VCC chains starting from it.

#### B.2.1 N-to-N Straight (Single Generator)

This configuration is used for throughput as well as latency measurements. The scalable versions can be obtained as follows:

a) Throughput measurements: For these tests, we need only a single chain starting from a single generator, i.e.,  $k=1$  and  $j=1$ . The chain starts from one port, goes through all other ports and exits from the starting port. Therefore,  $NP(1)$  is equal to  $N-1$ .  $C_{in}$  and  $C_{out}$  coincide and any port  $P_x$  could be selected to be the input/output port

Figure B.5 illustrates this case for the 2-module 8-port switch. Figure B.5a shows how to number the switch ports. Figure B.5b presents the VCC chain representation of the configuration. using  $C_{in}=C(0,1,1) = C_{out} = C(8,1,1) = P1$ .

## Proposed Appendix B of Performance Testing Baseline Text on Scalable Configurations

The application of the algorithm is simple. The ports  $C(i,1,1)$  in the VCC chain are selected in numerically increasing order. The ports are included in VCC chain if they are not already used up. After reaching Nth port, the index (i) starts again from the beginning (from  $i=1$ ).

Module 1	[1] P1	[2] P3	[3] P5	[4] P7
Module 2	[1] P2	[2] P4	[3] P6	[4] P8

Figure B.5a. Port numbering of a switch with 2 modules and 4 ports on each module. The numbers in brackets indicate the port numbers in the module.

For  $C_{in}=C_{out}=P1$ , the VCC chain is: P1-P2-P3-P4-P5-P6-P7-P8-P1

If we chose P3 as the source and destination then the VCC chain will be: P3-P4-P5-P6-P7-P8-P1-P2-P3

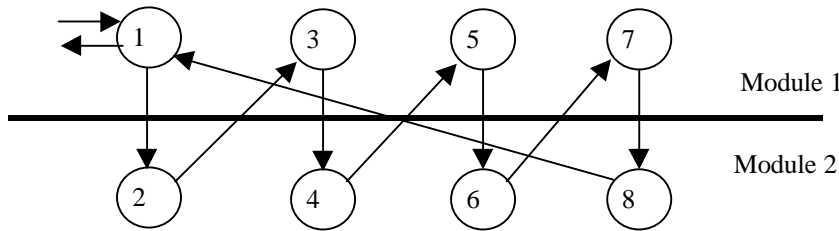


Figure B.5b. The 8-to-8 straight configuration with one generator.

Note that in both cases, the VCC chains cross the modules at every hop.

b) Latency Measurements: First, let us consider the case in which the background traffic does use the source/destination ports of the foreground traffic (but in the opposite direction). The background traffic passes through all other ports. Therefore,  $NP(1)$  is equal to  $N-2$ .  $C_{in}$  and  $C_{out}$  for the background coincide respectively with  $P_{Fout}$  and  $P_{Fin}$ . If  $P_{Fin}=P2$  and  $P_{Fout}=1$ , the foreground chain is P2-P1 and the background chain is P1-P2-P3-P4-P5-P6-P7-P8-P2. This connection configuration was presented earlier in Figures B.3 and B.4.

Now, let us consider the case in which the background traffic does not use the source/destination ports of the foreground. In this case,  $NP(1)$  is equal to  $N-3$ .  $C_{in}$  and  $C_{out}$  coincide and could be selected any of the switch ports  $P_x$  except  $P_{Fout}$  and  $P_{Fin}$ . For example, the foreground could use the chain P2-P1 and background could use P3-P4-P5-P6-P7-P8-P3. Figure B.6 illustrates this case.

## Proposed Appendix B of Performance Testing Baseline Text on Scalable Configurations

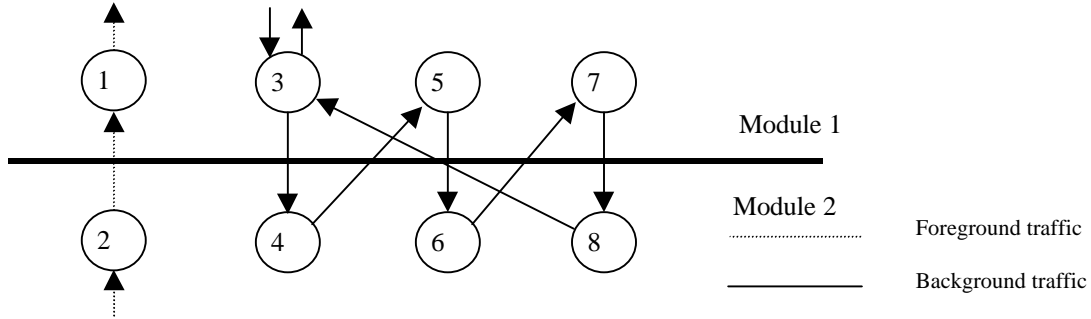


Figure B.6 Implementation of the 6-to-6 straight configuration with one generator.

### B.2.2. N-to-N Straight (r Generators)

This configuration implements the N-to-N straight configuration with  $r$  generators.

- a) Throughput Measurements: Each generator has one VCC chain. In all there are  $r$  VCC chains. Of the  $N$  ports,  $r$  ports are used as source/destination of these chains. The remaining ports are divided among the generators as evenly as possible.

Let  $p = \text{mod}(N-r, r)$

- For the first  $p$  VCC chains, the number of intermediate ports  $NP$  is equal to the quotient of  $(N-r)/r$  plus 1, i.e.,  $\lfloor (N-r)/r \rfloor + 1$
- For the remaining  $(r-p)$  VCC chains,  $NP$  is equal to the quotient of  $(N-r)/r$ , or  $\lfloor (N-r)/r \rfloor$
- For all VCC chains, the source/destination ports coincide and may be selected from any of the switch ports  $P_x$  not selected by other VCC chains as source or destination.

As an example, consider the 8-port switch again. With  $r=3$  generators,  $p$  equals  $\text{mod}(8-3, 3) = 2$ . So, the first two VCC chains have  $NP = \lfloor (8-3)/3 \rfloor + 1 = 2$  intermediate ports, and the last chain has  $NP = \lfloor (8-3)/3 \rfloor = 1$ .

Figures B.7 illustrates the implementation of the VCC chains for this case. First we select the source and destination ports:

Port 1 is the source and destination for the first chain, so  $C(0,1,1)=P1$ ,  $C(3, 1,1)=P1$

Port 2 is the source and destination for the second chain, so  $C(0,1,2)=P2$ ,  $C(3,1,2)=P2$

Port 3 is the source and destination for the third chain, so  $C(0,1,3)=P3$ ,  $C(2, 1,3)=P3$ .

These selections have been made to avoid any overlap.

Then we apply the algorithm. Let us start with the VCC chain having port 1 as the source. The next port available is  $P4$ , so  $C(1,1,1)=P4$ , then  $C(2,1,1)=P5$ . This VCC chain has two intermediate ports, for this reason it is now complete. Now we continue with the VCC chain starting at port 2. The next available port is port 6 (because 4 and 5 are fully occupied by the previous VCC chain). So  $C(1,1,2)=P6$ , and then  $C(2,1,2)=P7$ . Similarly,

**Proposed Appendix B of Performance Testing Baseline Text on Scalable Configurations**

$C(1,1,3)=P8$ . This VCC chain has only one intermediate port. The VCC chain implementation is complete.

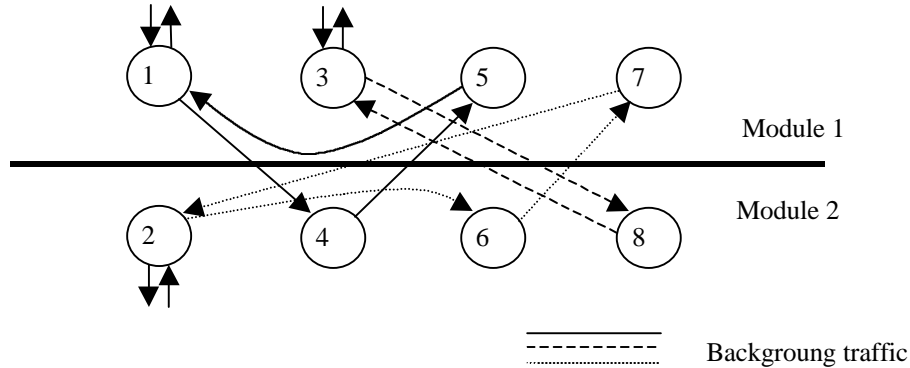


Figure B.7 Implementation of the 8-to-8 straight configuration with 3 generators .

b) Latency Measurements: Consider the case with the background traffic using the foreground ports in the opposite direction. The remaining  $N-1$  ports are evenly divided among the  $r$  background VCC chains.

Let  $p = \text{mod}(N-r-1, r)$

- For the first  $p$  VCC chains  $NP$  is equal to the quotient of  $(N-r-1)/r$  plus 1, i.e.,  $\lfloor (N-r-1)/r \rfloor + 1$
- For the remaining  $(r-p)$  VCC chains,  $NP$  is equal to the quotient of  $(N-r-1)/r$ , or  $\lfloor (N-r-1)/r \rfloor$
- For one of VCC chains,  $C_{in}$  and  $C_{out}$  coincide with  $P_{Fout}$  and  $P_{Fin}$  respectively.
- For the other VCC chains,  $C_{in}$  and  $C_{out}$  coincide and could be selected from any of the switch ports  $P_x$  not selected by other VCCs

Figure B.8 illustrates an example for this case. Ports 1 and 2 are used by the foreground traffic as destination and source ports, respectively.

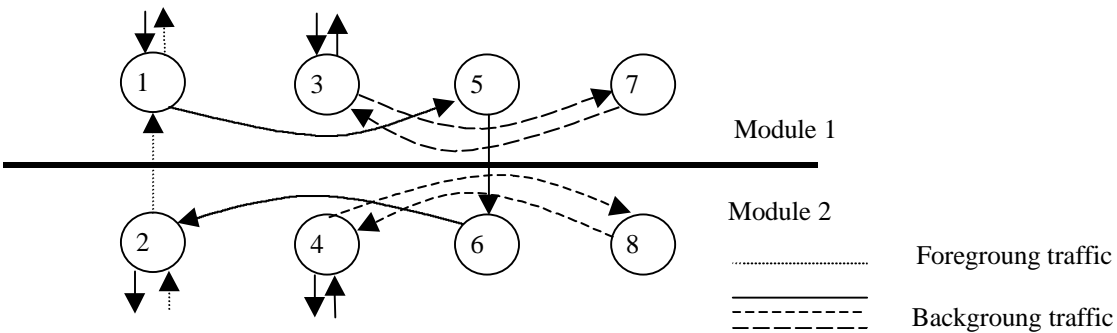


Figure B.8 Implementation of the 7-to-7 straight configuration with 3 generators for background traffic in latency measurement.

Ports 1 and 2 will be used as source and destination ports (respectively) by one of the background VCC chains. The other two generators will use port 3 and 4 as the source and

## Proposed Appendix B of Performance Testing Baseline Text on Scalable Configurations

destination ports, respectively. For the first VCC chain,  $NP = 2$  and for the other two VCC chains  $NP = 1$ . The chains are: P1-P5-P6-P2, P3-P7-P3, and P4-P8-P4. Note that the first chain goes from P1 to P5 since P2, P3, P4 since have already been assigned to other chains.

The configuration for the case when the background traffic does not share the ports with the foreground can be generated by the above procedure by considering the switch having only  $N-2$  ports

### B.2.3. N-to-m Partial Cross (r Generators)

This is a generalization of N-to-m Partial cross with 1 generator presented in the baseline. The discussion here applies for  $r=1$ . Also, by appropriately setting  $r$ , one can obtain non-scalable (basic) configurations.

a) Throughput Measurements: This configuration has  $m \cdot r$  VCC chains originating from  $r$ , where each generator originates  $m$  VCC chains. Each has a load of  $1/m^{\text{th}}$  of the generator. Each intermediate node has exactly  $m$  of these streams flowing through it. Again, the ports are evenly divided among the chains. However, since each chain uses only a part of the port's capacity, the ports can be used by other chains even from other generators as well.

Let  $p = \text{mod}(N-r, r)$

- For the first  $p$  VCC chains, the number of intermediate ports  $NP$  is equal to the quotient of  $(N-r)/r$  plus 1, i.e.,  $\lfloor (N-r)/r \rfloor + 1$
- For the remaining  $(r-p)$  VCC chains,  $NP$  is equal to the quotient of  $(N-r)/r$ , or  $\lfloor (N-r)/r \rfloor$
- For all  $m$  VCC chains, source/destination ports coincide and may be selected from any of the switch ports  $P_x$  not selected by other VCC chains.

Figure B.9 illustrates the case of 8-to-2 partial cross with 2 generators.

In this case,  $p = \text{mod}(8-2, 2) = 0$ . So, the VCC chains of both generators have  $\lfloor (8-2)/2 \rfloor = 3$  intermediate ports.

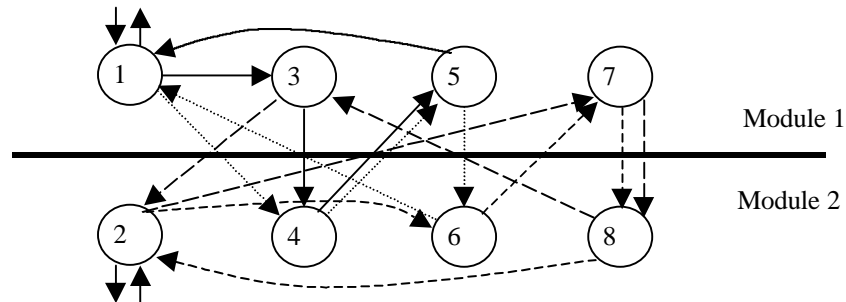


Figure B.9 Implementation of 8-to-2 partial cross configuration with 2 generators for foreground traffic

Both of the VCC chains of the first generator start and end at port 1, so:

## Proposed Appendix B of Performance Testing Baseline Text on Scalable Configurations

$$C(0,1,1)=C(0,2,1)= C(4,1,1)=C(4,2,1)=P1.$$

Similarly for the two VCC chains of the other generator:

$$C(0,1,2)=C(0,2,2)=C(4,1,2)=C(4,2,2)=P2.$$

First we divide the remaining ports among the two generators. The first generator gets P3, P4, and P5. The second generator gets P6, P7, and P8.

The first chain of the first generator is simply P1-P3-P4-P5-P1.

The first chain of the second generator is P2-P6-P7-P8-P2. The second chain from the first generator is obtained by shifting the intermediate ports of the first chain. Therefore, the chain is P1-P4-P5-P6-P1. Note that this chain is sharing port P6 of the other generator since each chain uses only half the capacity. The second chain of the second generator is again obtained by shifting: P2-P7-P8-P3-P2. Note that, shifting P8 would have produced P1 but P1 is being fully used. The next port P2 is also being fully used. So P3 is used.

b) Latency measurements: Again we consider only the case of background traffic sharing the foreground ports in the opposite direction. Excluding the foreground port, the remaining  $N-1$  ports are evenly divided among the  $r$  generators.

Let  $p = \text{mod}(N-r-1, r)$

- For all VCCs of the first  $p$  generators  $NP$  is equal to the quotient of  $(N-r)/r$  plus 1, i.e.,  $\lfloor (N-r)/r \rfloor + 1$
- For all VCCs of the remaining  $(r-p)$  generators,  $NP$  is equal to the quotient of  $(N-r)/r$ , or  $\lfloor (N-r)/r \rfloor$
- For all  $m$  VCCs of only one generator, source and destination coincide with  $P_{\text{Fout}}$  and  $P_{\text{Fin}}$  respectively.
- For all  $m$  VCCs of all other generators, source and destination coincide and could be selected from any of the switch ports  $P_x$  not selected by other generators.

An example of this case is shown in Figure B.10. In this case,  $N=8$ ,  $r=2$ . So  $p = \text{mod}(8-2-1, 2) = 1$ , so  $NP(1)=3$  and  $NP(2)=2$ .

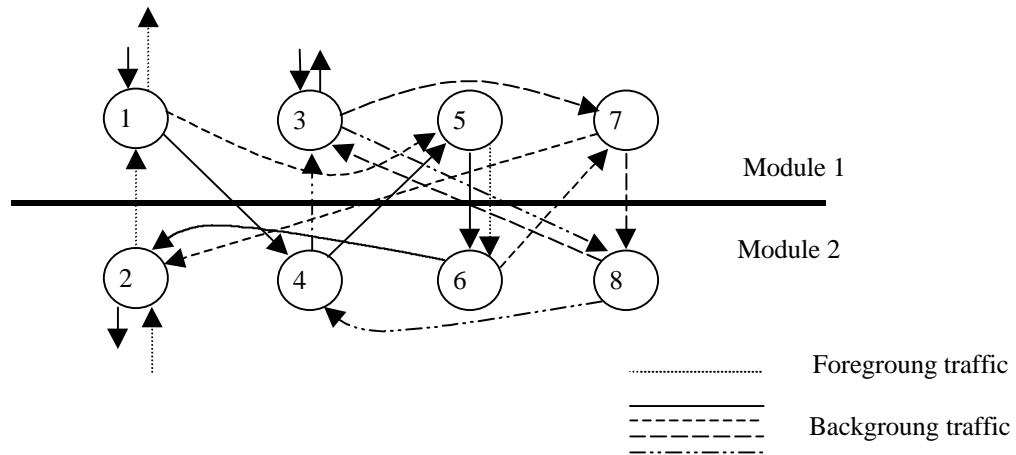


Figure B.10 Implementation of 7-to-2 partial cross configuration with 2 generators for background traffic in latency measurements.

## Proposed Appendix B of Performance Testing Baseline Text on Scalable Configurations

The VCC chains of the first generator will use ports 1 and 2 in opposite directions of the foreground traffic,. The VCC chains of the second generator will use port 3 as the source and destination. The chains of the first generator are: P1-P4-P5-P6-P2, P1-P5-P6-P7-P2. The chains of the second generator are: P3-P7-P8-P3, P3-P8-P4-P3.

Table B.1 summarizes the values for number of intermediate ports in various configurations of this section **B.2**. These values are used in the pseudocode of Section **B.3**.

	<b>N-to-N straight (Single generator)</b>	<b>N-to-N straight (r Generators)</b>	<b>N-to-m Partial Cross (Single Generator)</b>	<b>N-to-m Partial Cross (r Generators)</b>
<b>Number of Intermediate ports NP</b>	<b>See B.2.1.</b> a) N-1 b) N-2 c) N-3	<b>See B.2.2.</b> $\lfloor (N-r)/r \rfloor + 1$ or $\lfloor (N-r)/r \rfloor$	<b>See B.2.3.</b> a) N-1 b) N-2	<b>See B.2.3.</b> $q((N-r)/r) + 1$ or $\lfloor (N-r)/r \rfloor$

Table B.1 Parameter values used in the algorithm to creating VCC chains for different configurations.

**B 3. Algorithm for creating VCC Chains.**

The algorithm for creating VCC chains for different connection configuration is based on the definitions given in section **B.1.** and the characteristics specified in section **B.2** and summarized in Table B.1.

- **NP(k)** denotes the number of intermediate ports for the VCC chains of the  $k^{\text{th}}$  generator. These values are specified in **B.2.**
- **P(f)** denotes the  $f^{\text{th}}$  port of the switch
- **C(i, j, k)** denotes the  $i^{\text{th}}$  intermediate port of the  $j^{\text{th}}$  VCC chain of the  $k^{\text{th}}$  generator
- The function  $\text{mod}^*(x, N)$  is equal to  $\text{mod}(x, N)$  except for the cases where  $\text{mod}(x, N)$  is equal to zero, where the function is equal to  $N$

```

f = 1
For k = 1 to r, step 1
{ if(k>1){

    
$$f = 1 + \sum_{d=1}^{k-1} NP(d)$$


}

for j = 1 to m, step 1
{
    if (r is equal to 1 and j > 1)  $f = \text{mod}^*(f+1, N)$ 
    if (r>1 and j>1)  $f=C(2,j-1,k)$ 
    for i=1 to NP(k), step 1
    {
        while (P(f) is used as source or output by another VCC chain or is full)
        ↓ {
             $f = \text{mod}^*(f+1, N)$ 
        }

         $C(i, j, k) = P(f)$ 
         $f = \text{mod}^*(f+1, N)$ 
    }
} end for i

} end for j

} end for k.

```