DCQCN Fluid Model

1. **Fluid Model**

The DCQCN fluid model is as follows:

 (1)

 (2)

 (3)

 (4)

 (5)

 are in packets/second.  is the CNP generation interval (default 50us). is the recover interval (default 55us). *B* is the byte counter threshold in packets. *F* is the # of fast recovery. *N* is the # of flows. *C* is the bottleneck’s bandwidth. are the marking thresholds on switches. *q* is the queue length.  is the delay of the CNP arrival after the queue is congested.

Because the NIC sends CNPs at fixed interval , for simplicity we use the average delay for  (*D* is the propagation delay; no queuing delay given modern switches mark ECN on dequeuing):

 (6)

1. **Fixed Point**

Below for simplicity we denote:

 (7)

For the fixed point, we can simplify the equations (1) and (2):





We get the following unique fixed point:

 (8)

 (9)

Equation (1) and (2) each gives a value for , they are:

 (10)

And:

 (11)

Combine (10) and (11) we get the equation that determines *p*:

 (12)

**2.1 Numerical solution**

We use numerical tools to solve above equation with typical parameter setting

 (13)

We get:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| *p* | 0.04% | 0.05% | 0.06% | 0.07% | 0.08% | 0.08% | 0.09% | 0.09% | 0.10% | 0.10% |
| N | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |  |
| *p* | 0.11% | 0.11% | 0.12% | 0.12% | 0.13% | 0.13% | 0.14% | 0.14% | 0.15% |  |

Assuming, we get the fixed point of q(t): 34.25KB.

**2.2 Analytic approximation**

From (12), we obtain the taylor series around p=0 of the left hand side:



If we omit the *O(p4)* term, we have the fixed point:








1. **Stability Analysis**

**3.1 Linearization**

First Use Taylor series to approximate p

, , ,

, 









Denote, we linearized this system:

,,  , 

We have:









**3.2 Laplace Transform**









Use *RC(s)* to describe all other variables.



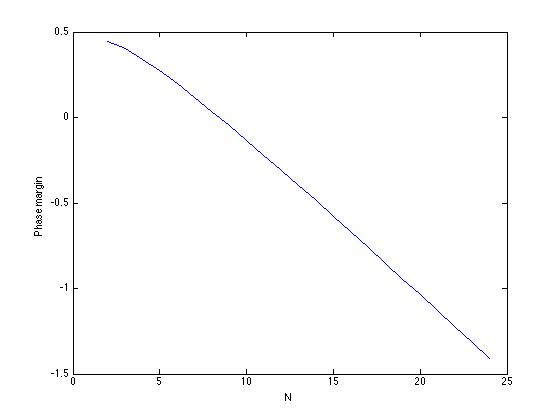




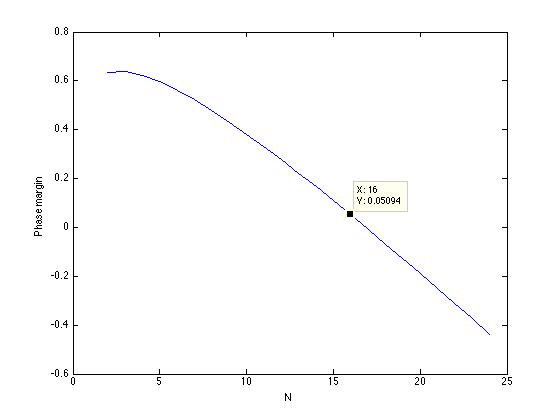
Finally:



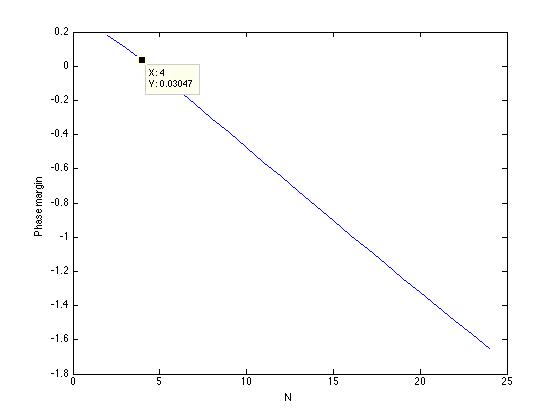
Sweep # of flows (default parameter):



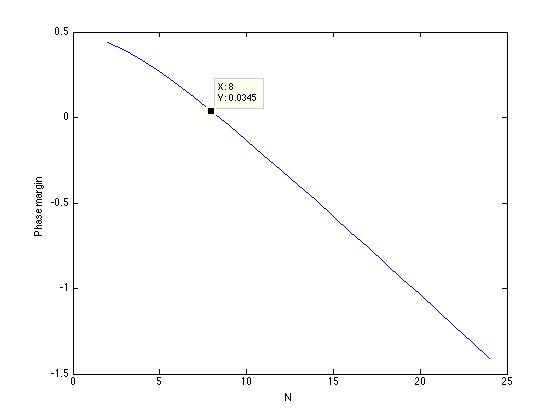
Sweep # of flows (half R\_ai):



Sweep # of flows (double timer):



Sweep # of flows (10x byte counter):



(Simplify a bit) Assuming p(t) does not change

We have linearized equations:





(notes for mathematica)

a == 1-(1-p)^(t\*C/N)

b == p/((1-p)^(-B)-1)

c == p\*(1-p)^(F\*B)/((1-p)^(-B)-1)

d == p/((1-p)^(-T\*C/N)-1)

e == p\*(1-p)^(F\*T\*C/N)/((1-p)^(-T\*C/N)-1)

\alpha = (1-(1-p)^(t’\*C/N))

a\*a\*\alpha/((b+d)\*(c+e)) == \tau\*\tau\*R\*C/N

Solve[{a == 1-(1-p)^(t\*C/N), a\*a\*(1-(1-p)^(t’\*C/N))/((p/((1-p)^(-B)-1)+ p/((1-p)^(-T\*C/N)-1))\*(p\*(1-p)^(F\*B)/((1-p)^(-B)-1)+ p\*(1-p)^(F\*T\*C/N)/((1-p)^(-T\*C/N)-1))) == t\*t\*R\*C/N}, {p}]

(1-(1-p)^(t\*C/N))^2

((1-(1-p)^(t\*C/N))^2)\* (1-(1-p)^(t’\*C/N))/((p/((1-p)^(-B)-1)+ p/((1-p)^(-T\*C/N)-1))\*(p\*(1-p)^(F\*B)/((1-p)^(-B)-1)+ p\*(1-p)^(F\*T\*C/N)/((1-p)^(-T\*C/N)-1))) == t\*t\*R\*C/N

Series[((1-(1-p)^(t\*C/N))^2)\* (1-(1-p)^(t’\*C/N))/((p/((1-p)^(-B)-1)+ p/((1-p)^(-T\*C/N)-1))\*(p\*(1-p)^(F\*B)/((1-p)^(-B)-1)+ p\*(1-p)^(F\*T\*C/N)/((1-p)^(-T\*C/N)-1))),{p,0,2}]