What does the KCC do?

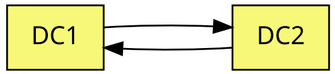
Douglas Bagnall

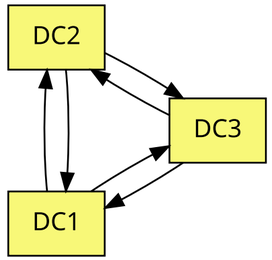
douglas.bagnall@catalyst.net.nz dbagnall@samba.org

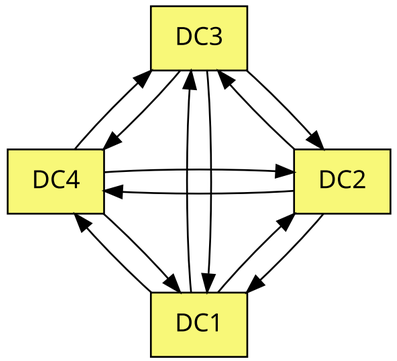
The **K**nowledge **C**onsistency **C**hecker creates a replication graph linking Domain Controllers.

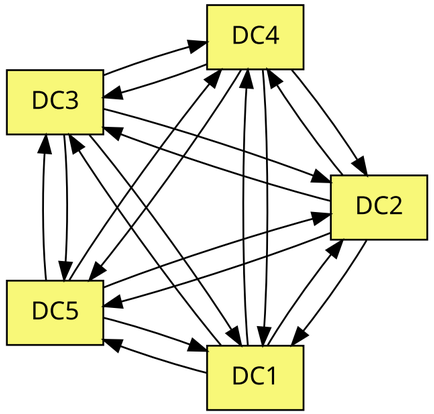
- barely changed since Windows 2000.

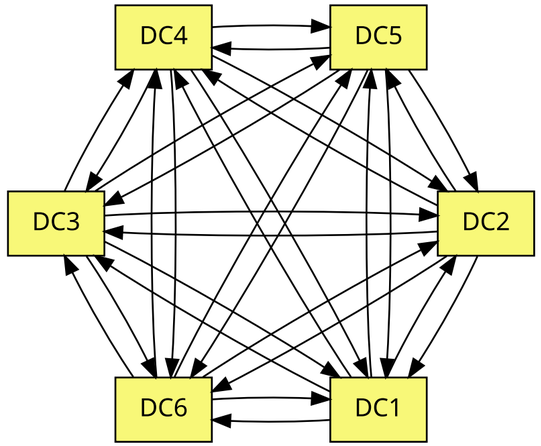
- we have samba_kcc since Samba 4.3 (2015).

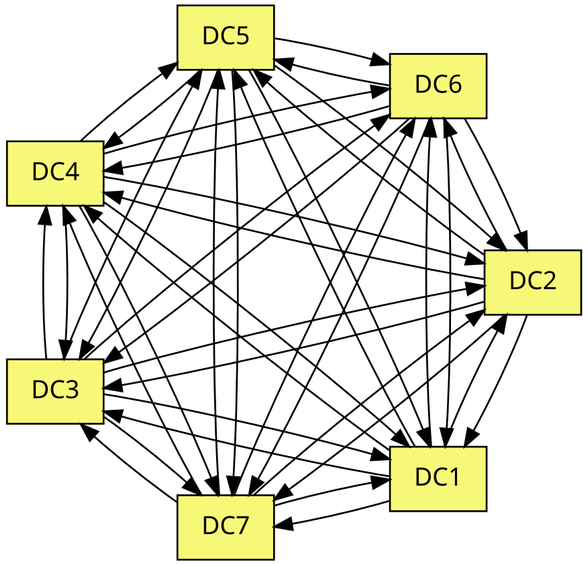

DC1
DC2

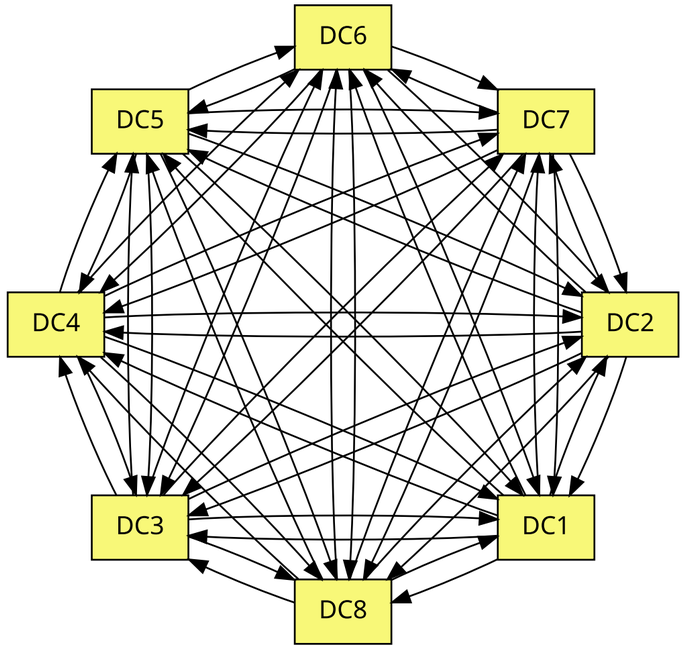

DC2
DC3
DC1

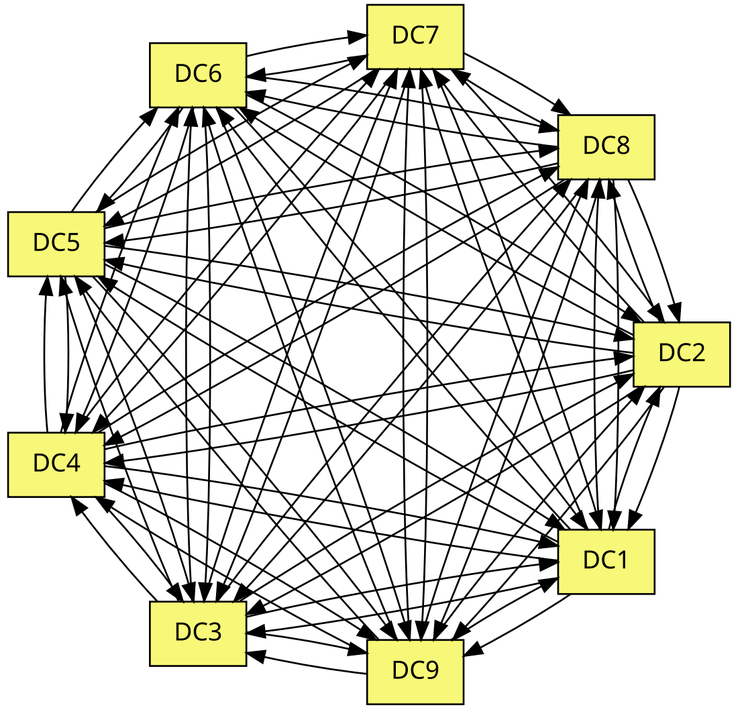

DC3
DC4
DC2
DC1

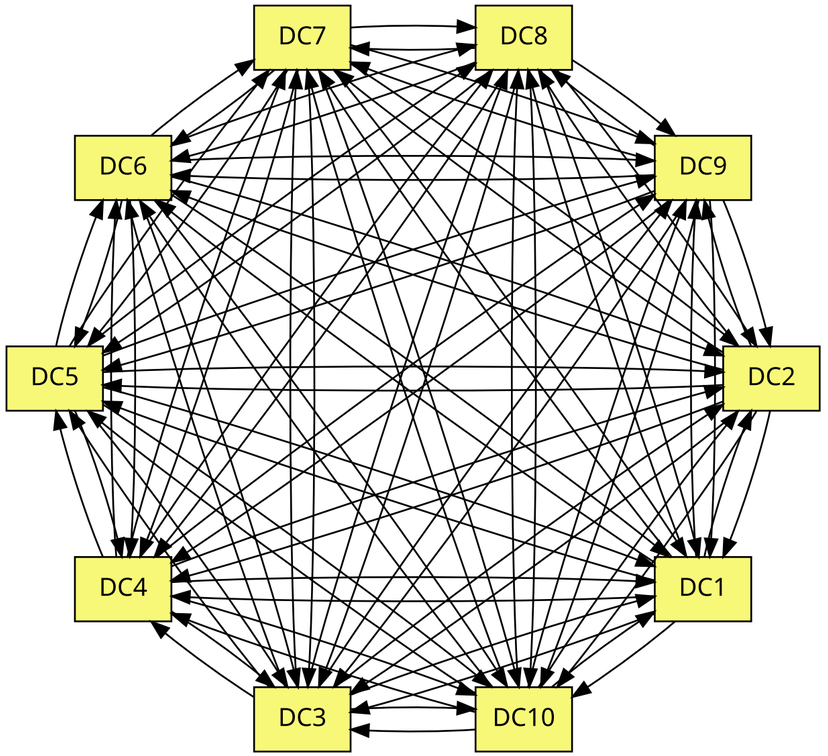

DC4
DC3
DC2
DC5
DC1

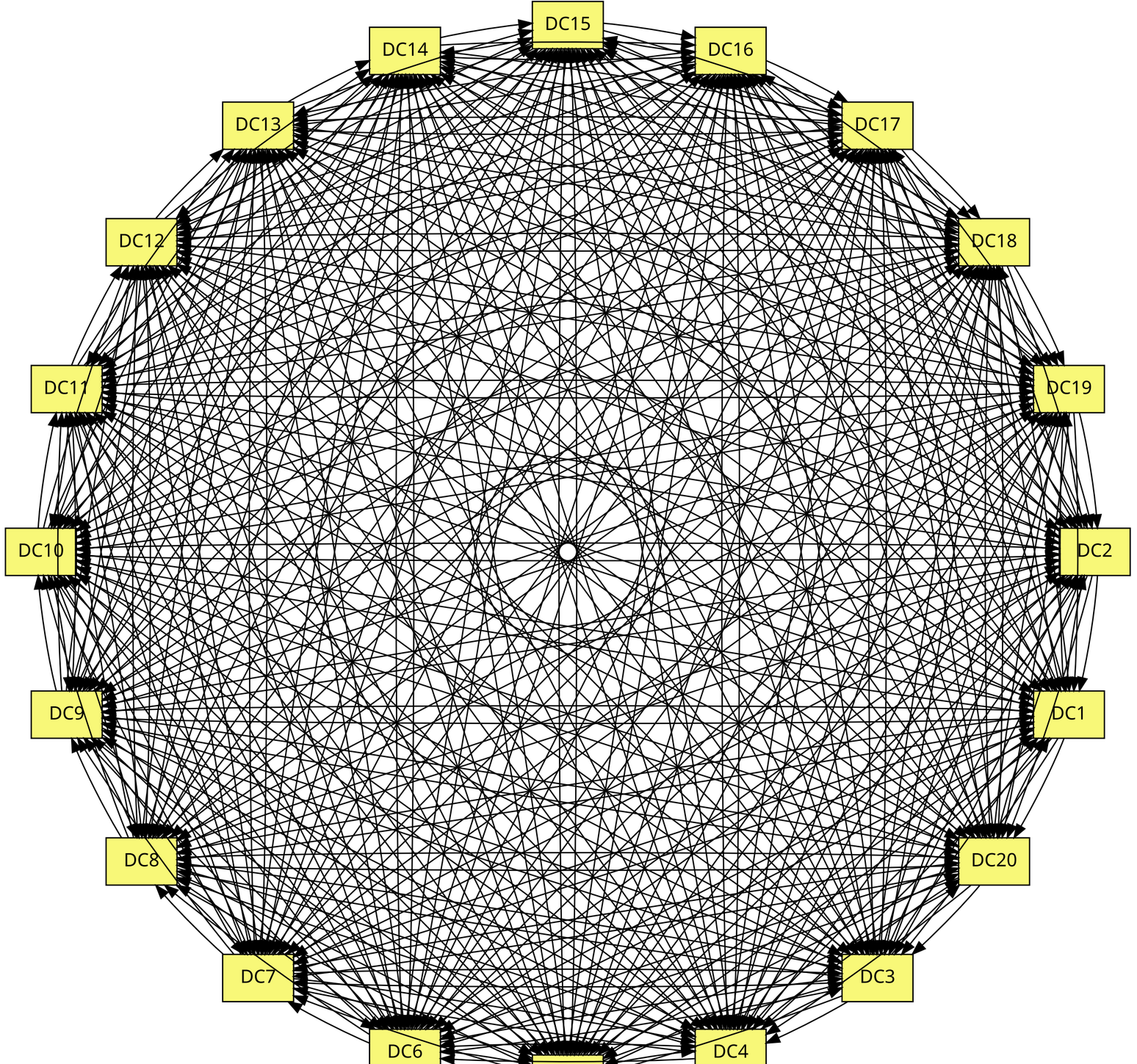

DC4
DC5
DC3
DC2
DC6
DC1

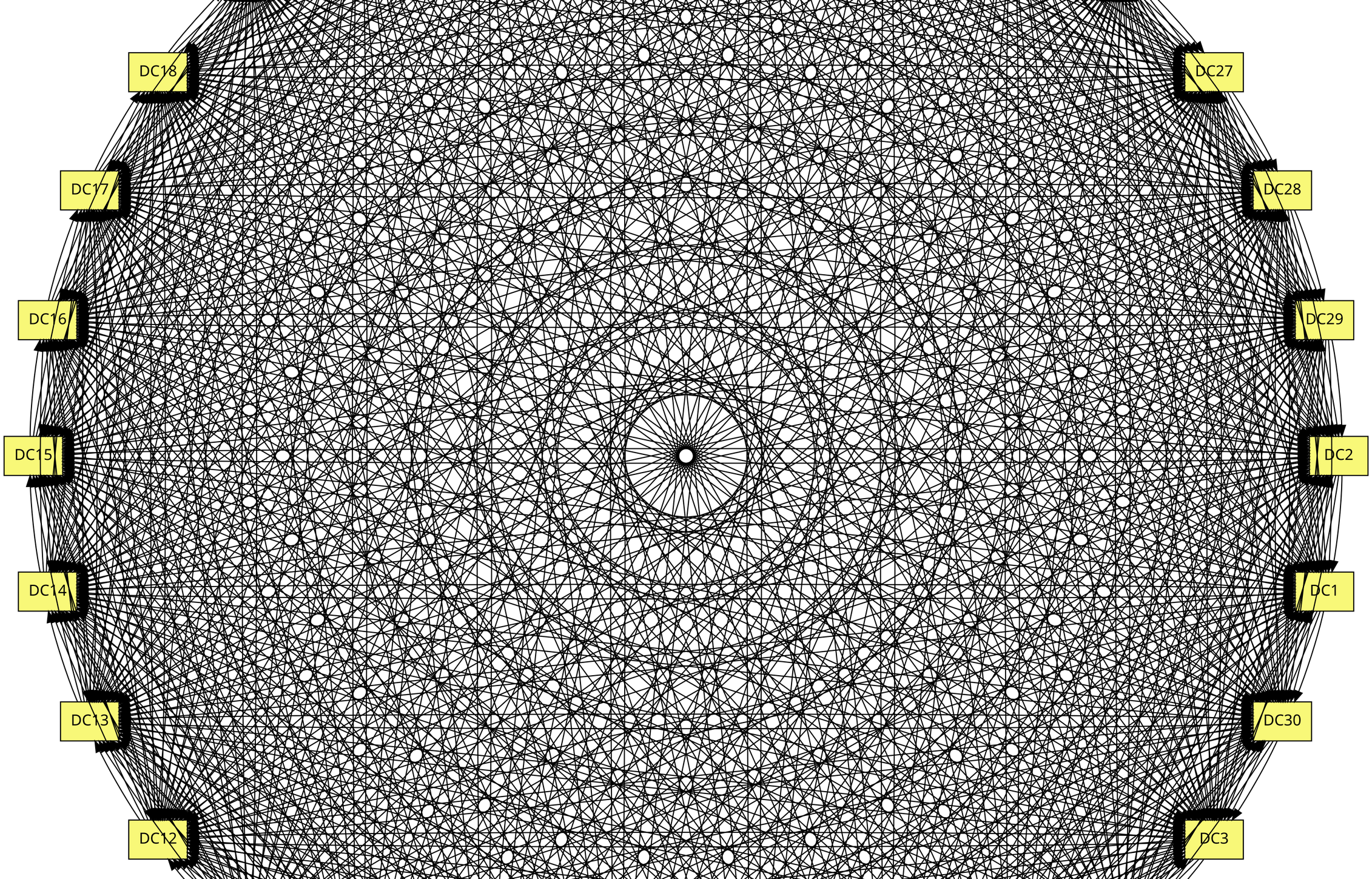

DC5
DC6
DC4
DC2
DC3
DC1
DC7

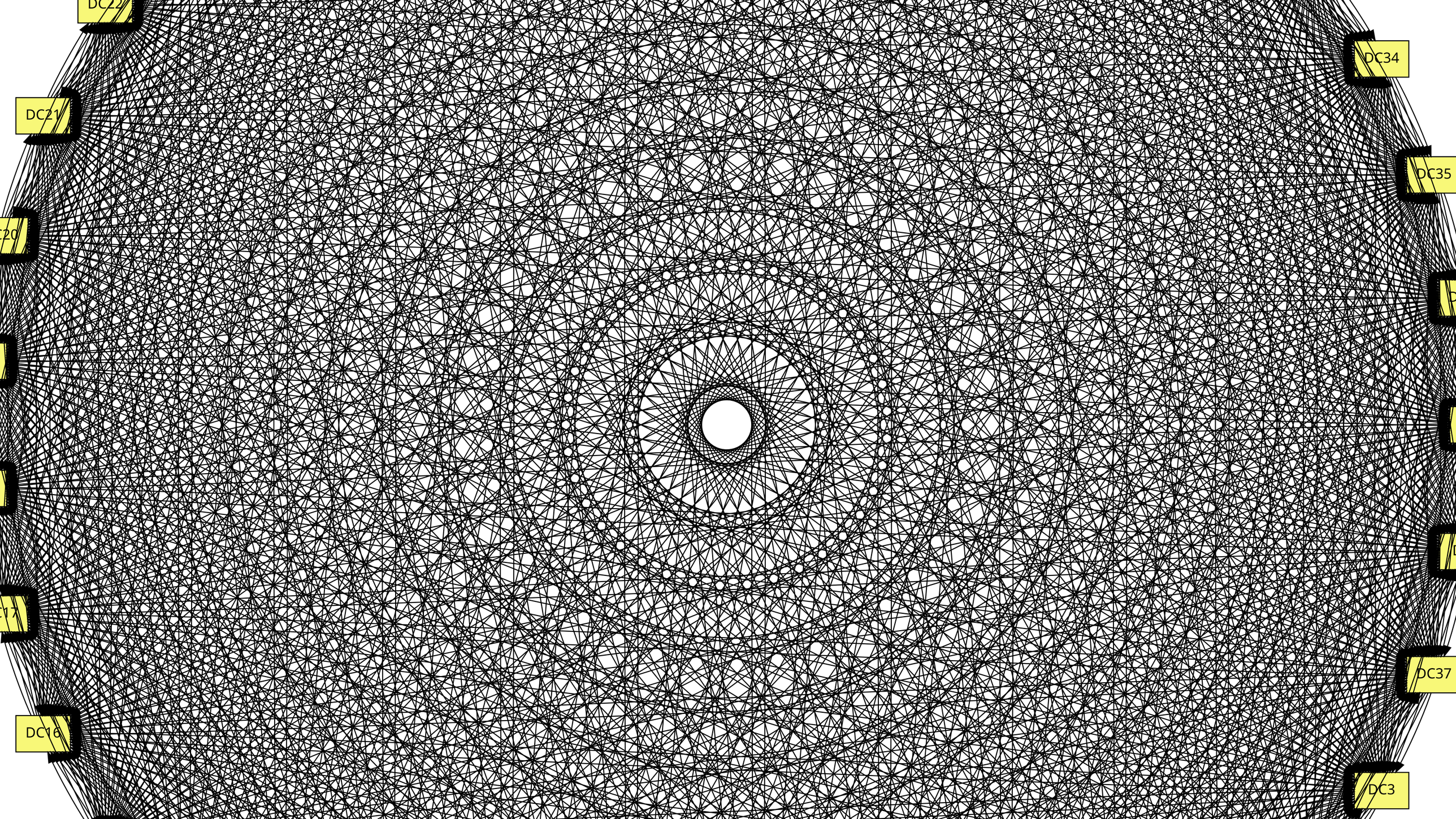

DC6
DC5
DC7
DC4
DC2
DC3
DC1
DC8

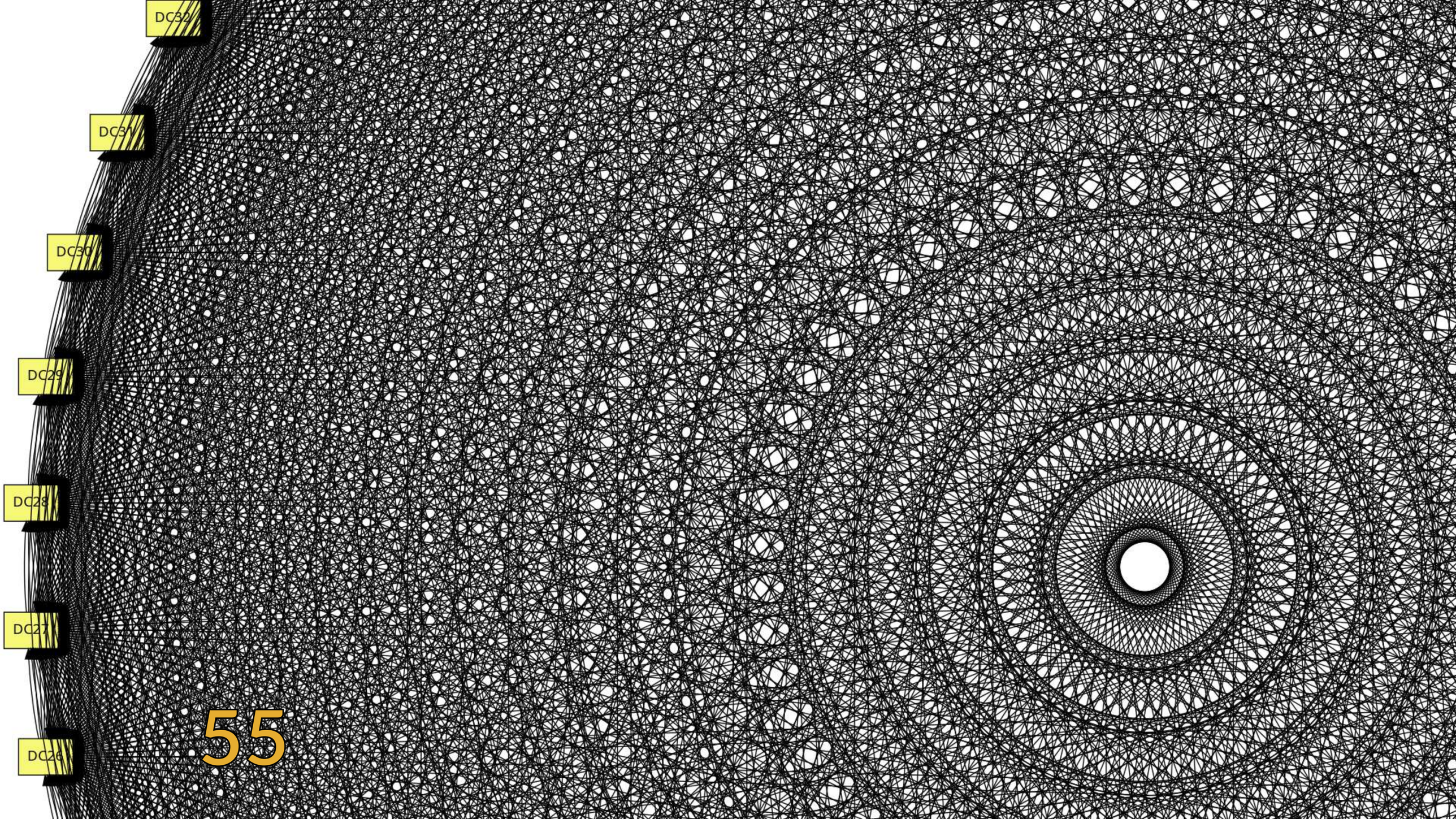

DC7
DC6
DC8
DC5
DC2
DC4
DC1
DC3
DC9

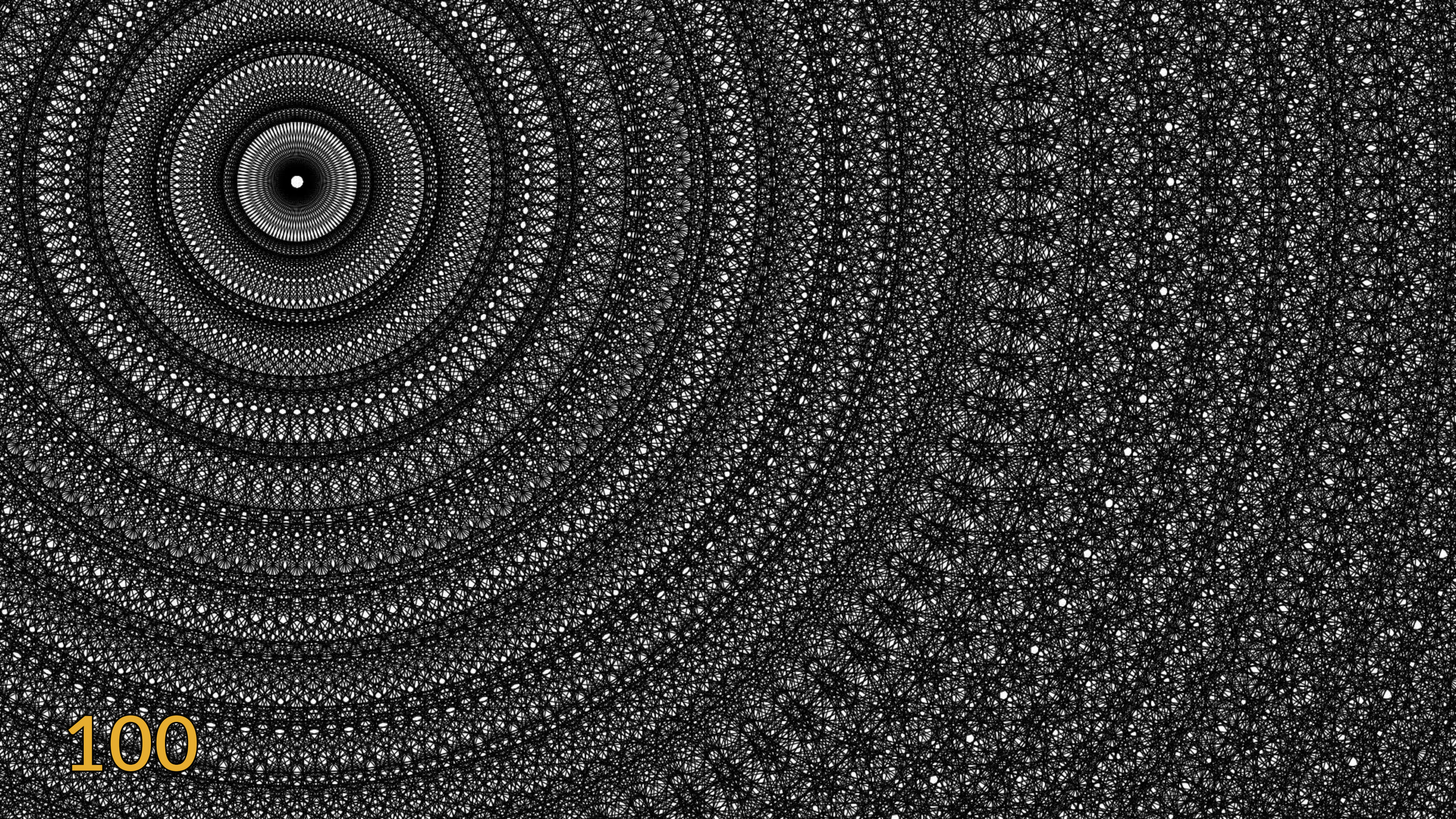

DC7
DC8
DC6
DC9
DC5
DC2
DC4
DC1
DC3
DC10


DC15
DC14
DC16
DC13
DC17
DC12
DC18
DC11
DC19
DC10
DC2
DC9
DC1
DC8
DC20
DC7
DC3
DC6
DC4

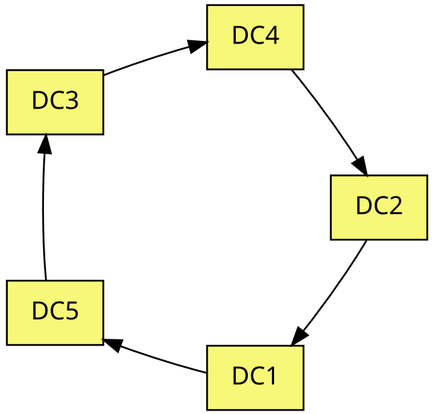

DC18
DC27
DC17
DC28
DC16
DC29
DC15
DC2
DC14
DC1
DC13
DC30
DC12
DC3


DC34
DC21
DC35
DC37
DC16
DC3

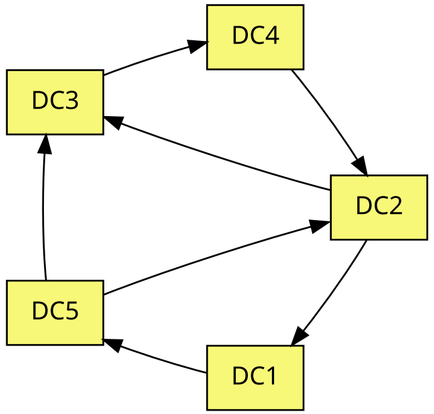

DC32
DC31
DC30
DC29
DC28
DC27
DC26
55


100

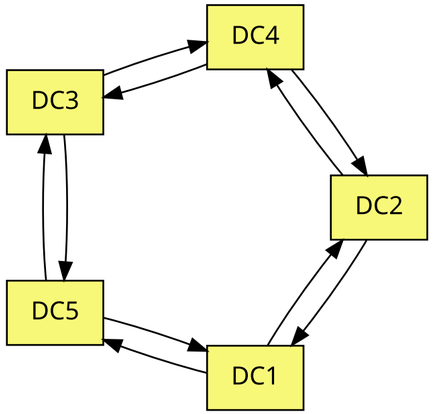
Minimal connected graph



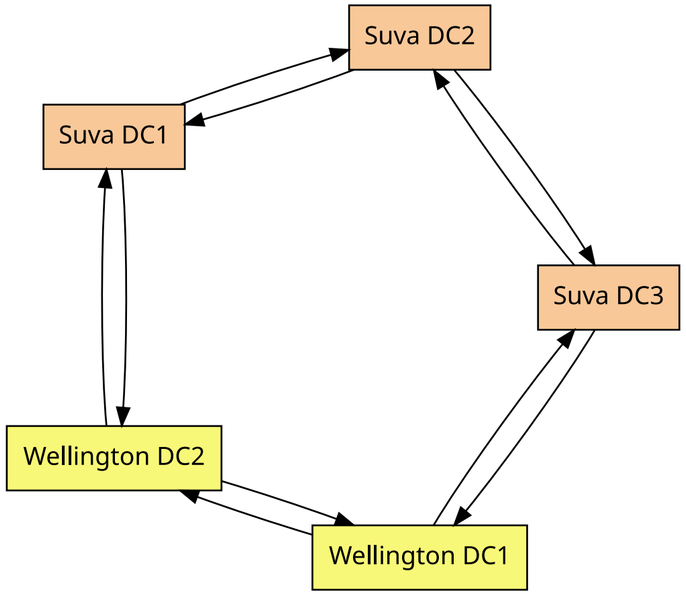
Minimal connected graph with shortcuts

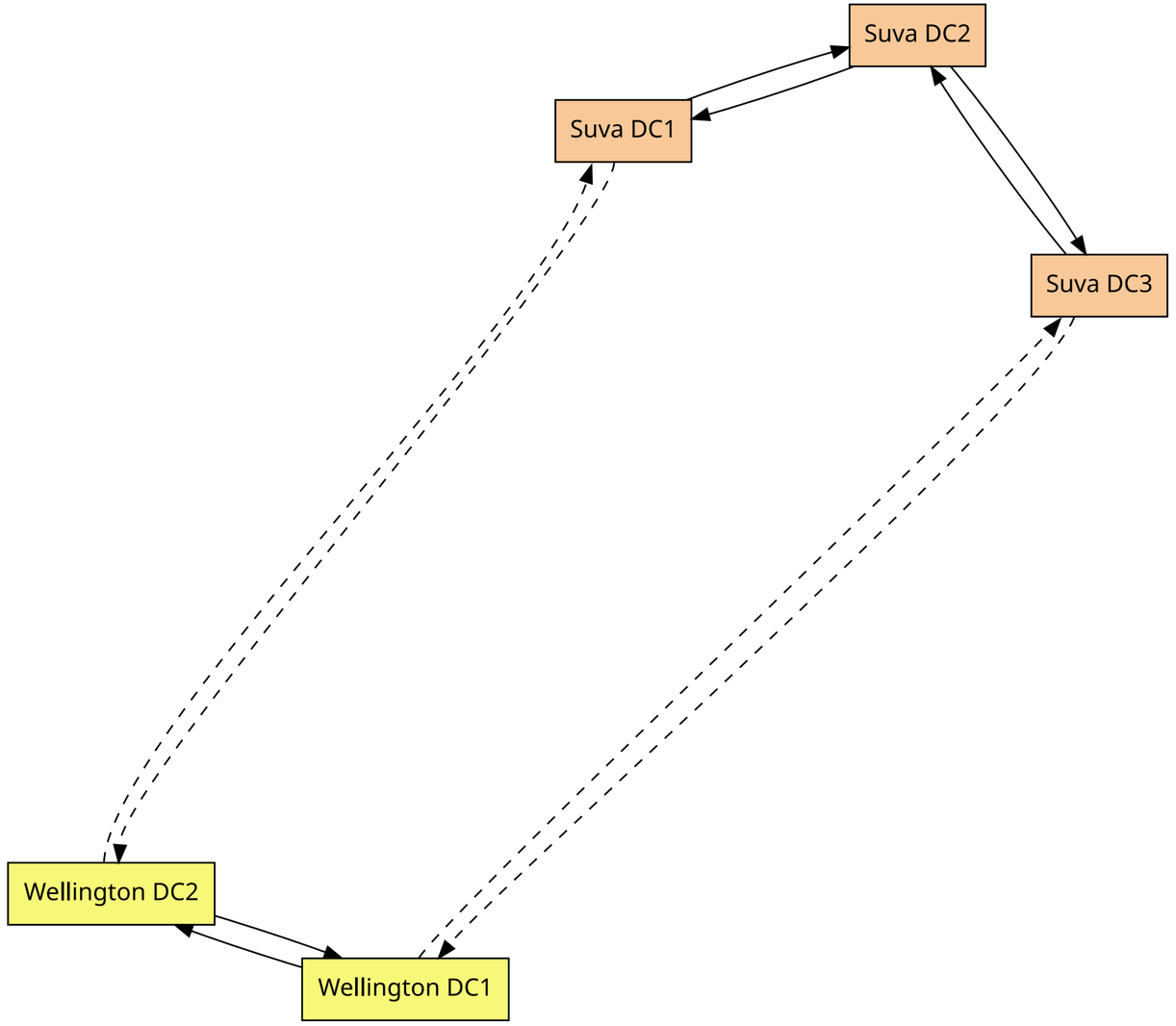


double linked graph

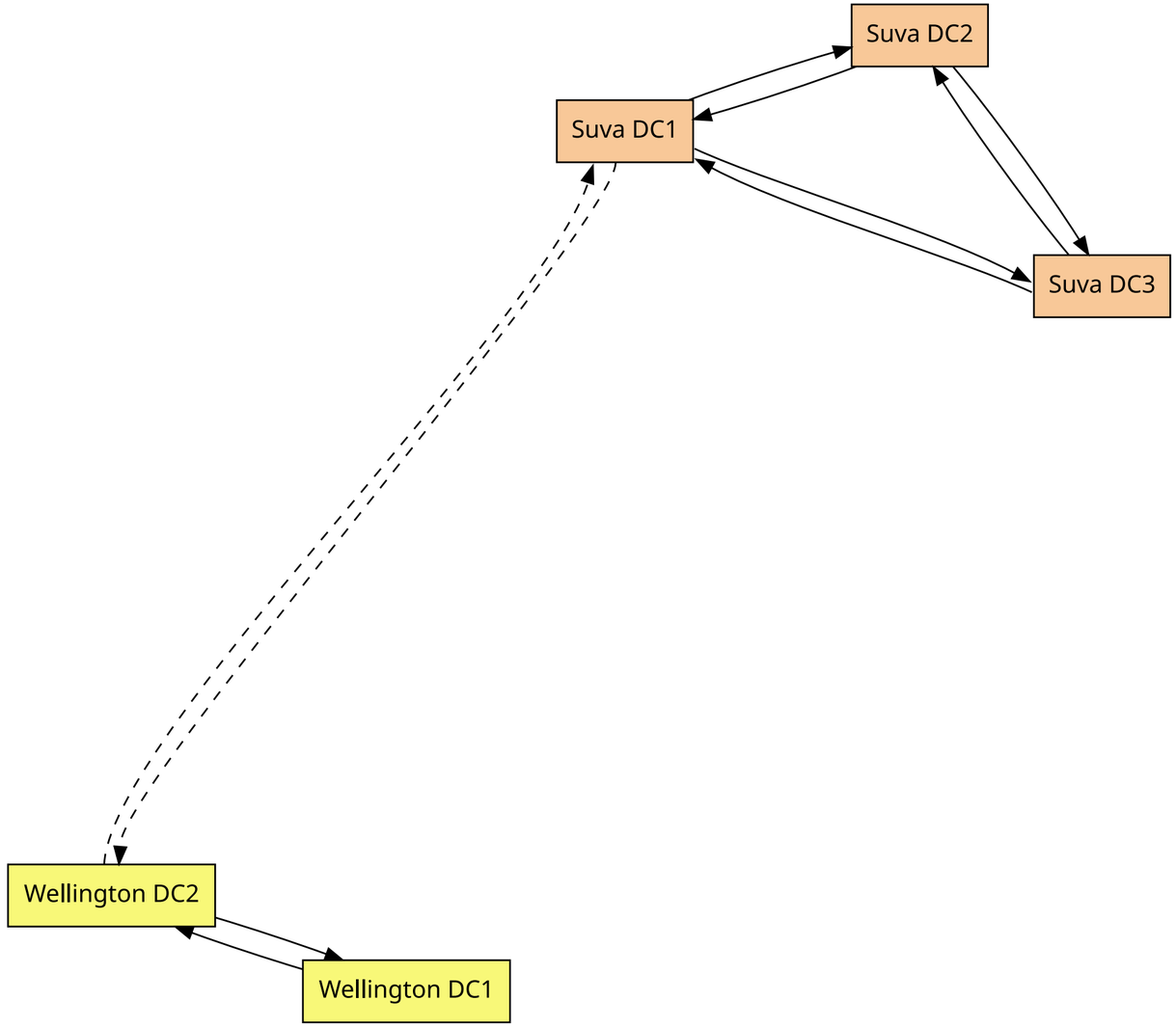


two sites

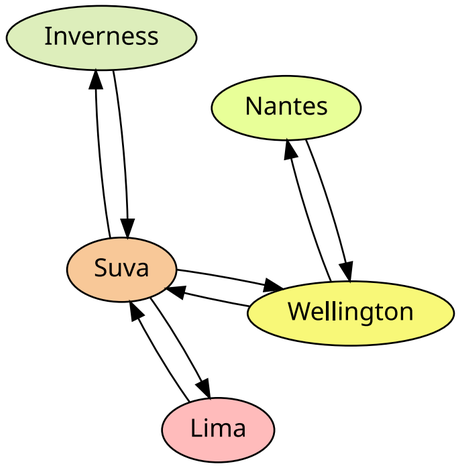



Suva DC2
Suva DC1
Suva DC3
Wellington DC2
Wellington DC1

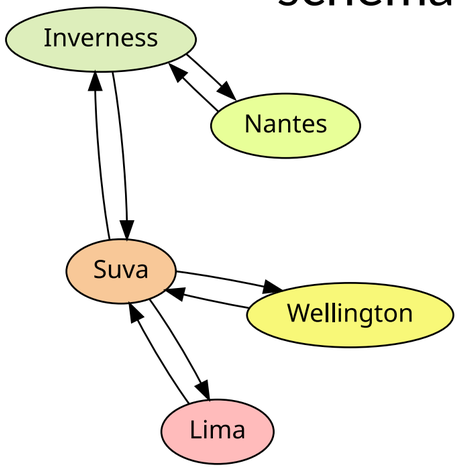
KCC does this



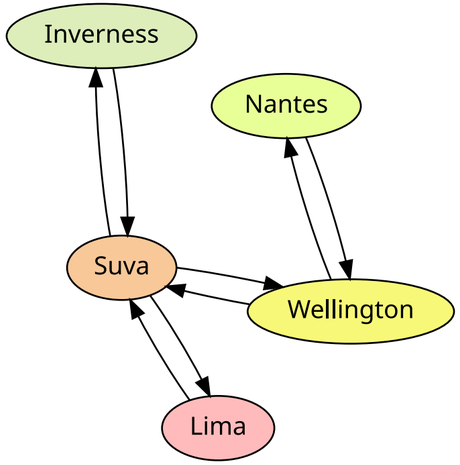
intersite graph is a *tree*



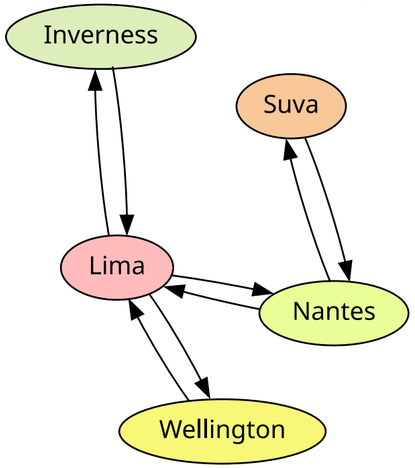
schema

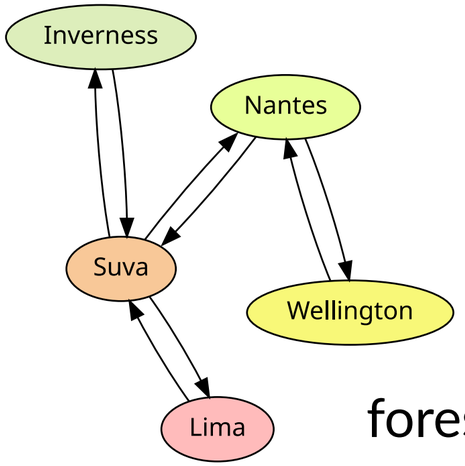


domain

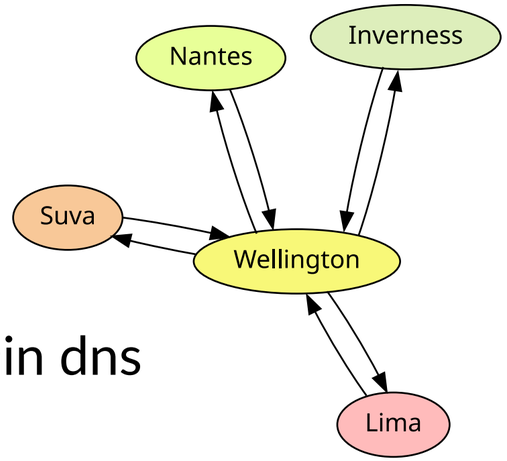


configuration

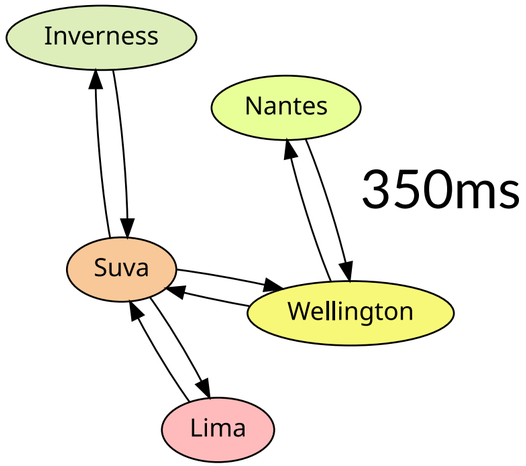


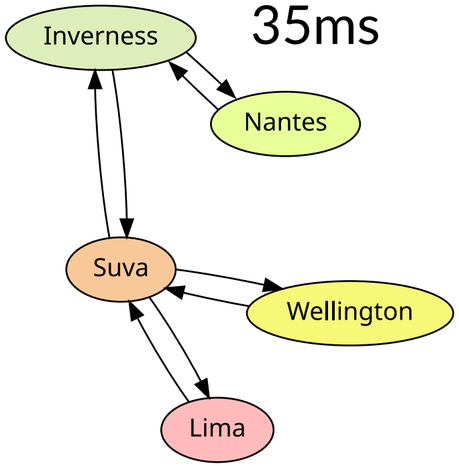


forest dns

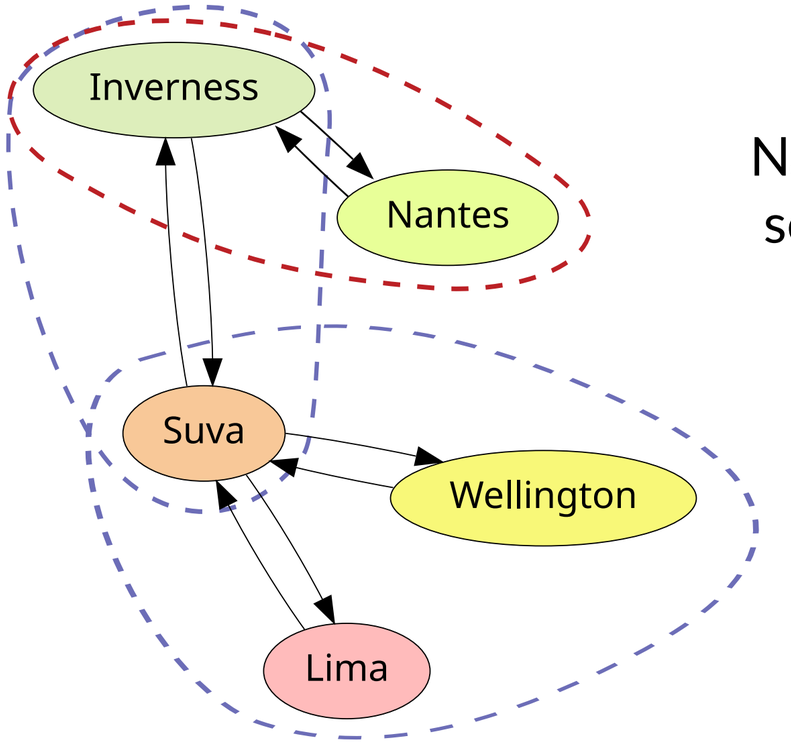


domain dns


Inverness
Nantes
350ms
Suva
Wellington
Lima

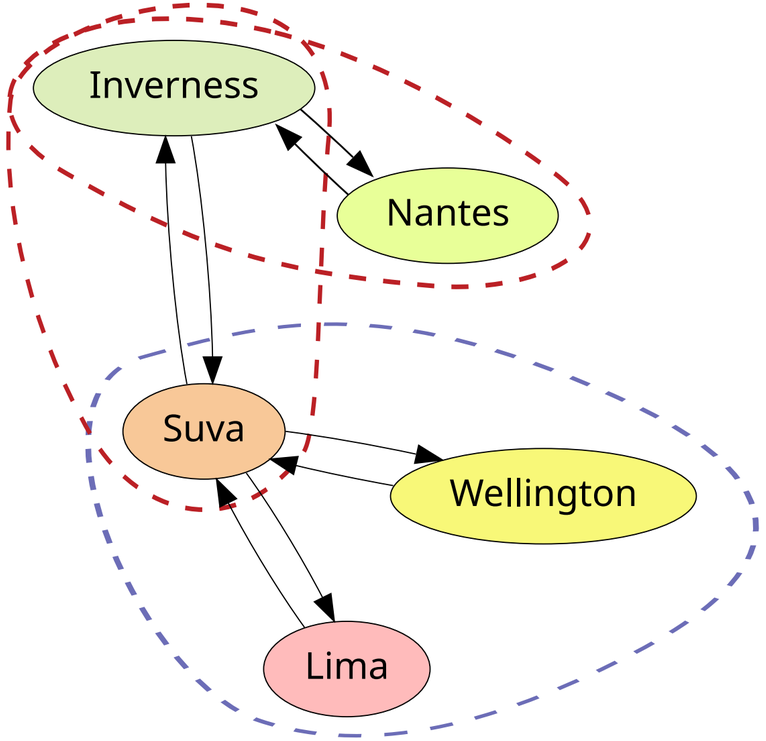

Inverness
35ms
Nantes
Suva
Wellington
Lima

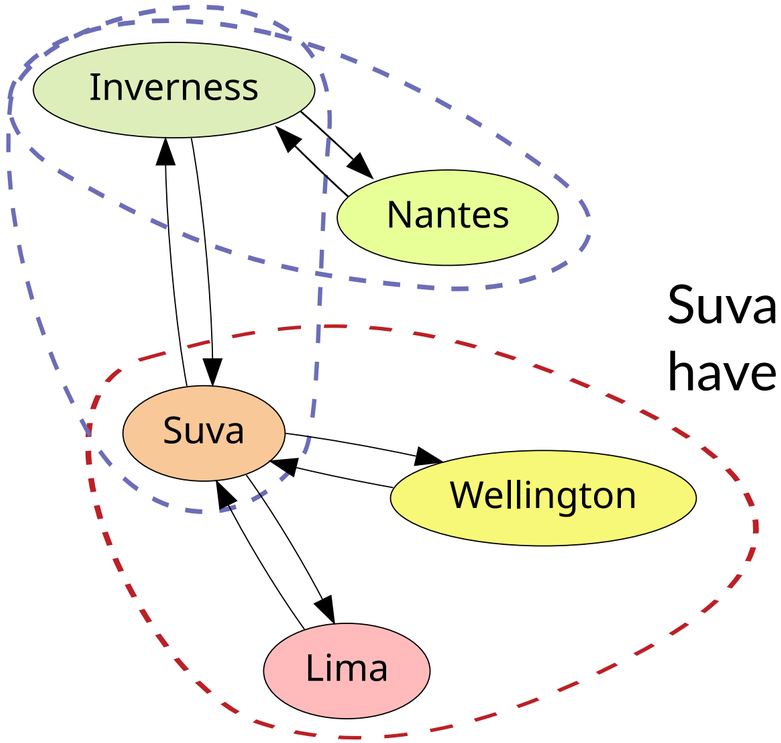
Site Links



Nantes can only see Inverness

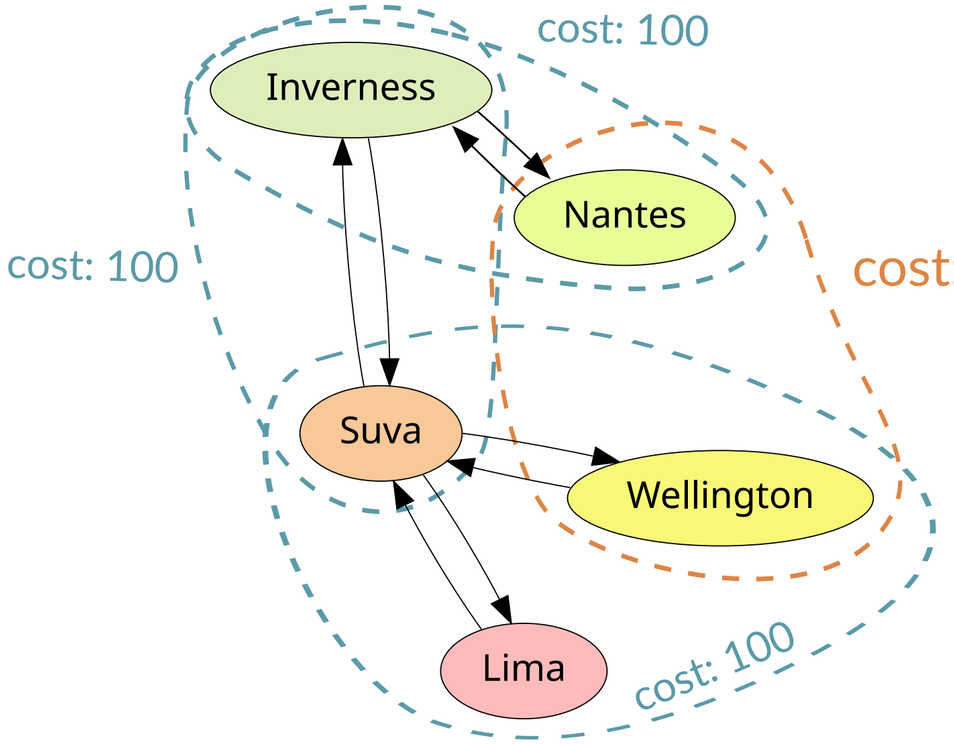
Inverness can see
Nantes and Suva





Suva, Wellington, and Lima have free reign to form a tree

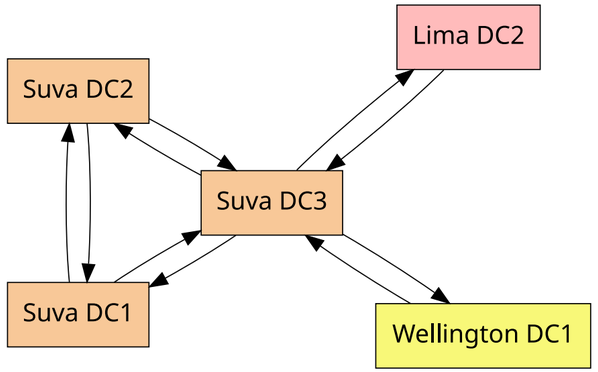
Site links can have a *cost*

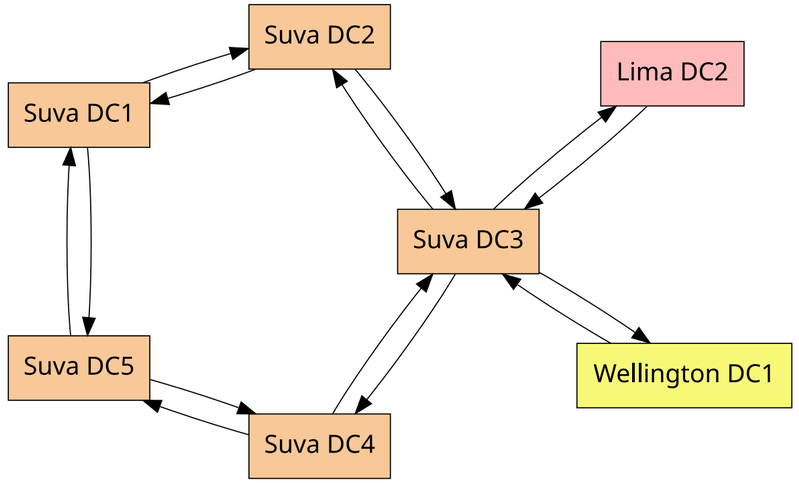


“I exist but pretend I don’t”

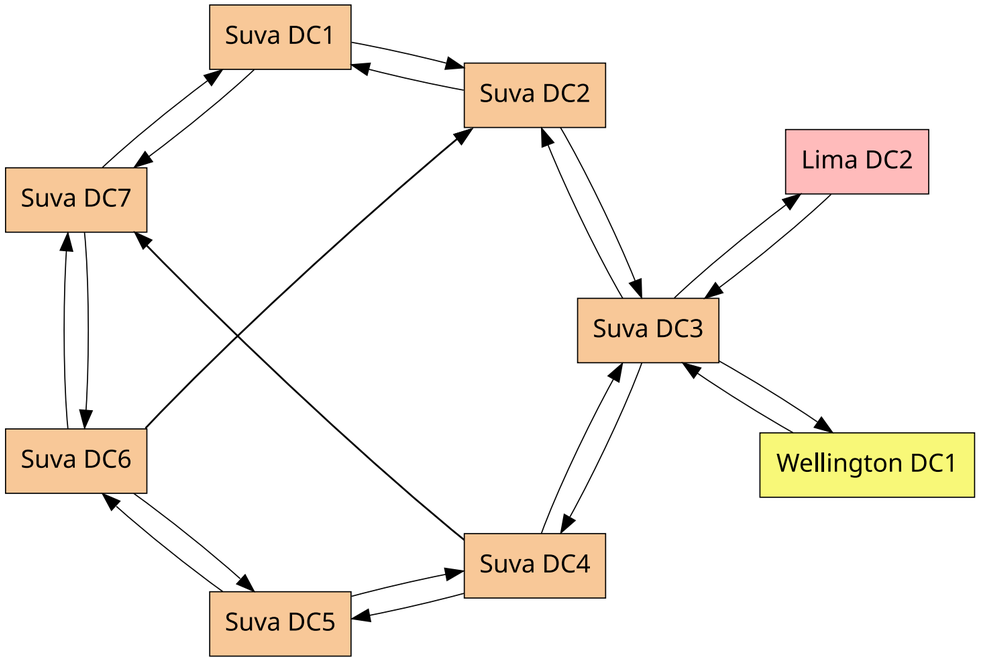
Intra-site links

double-linked ring in GUID order

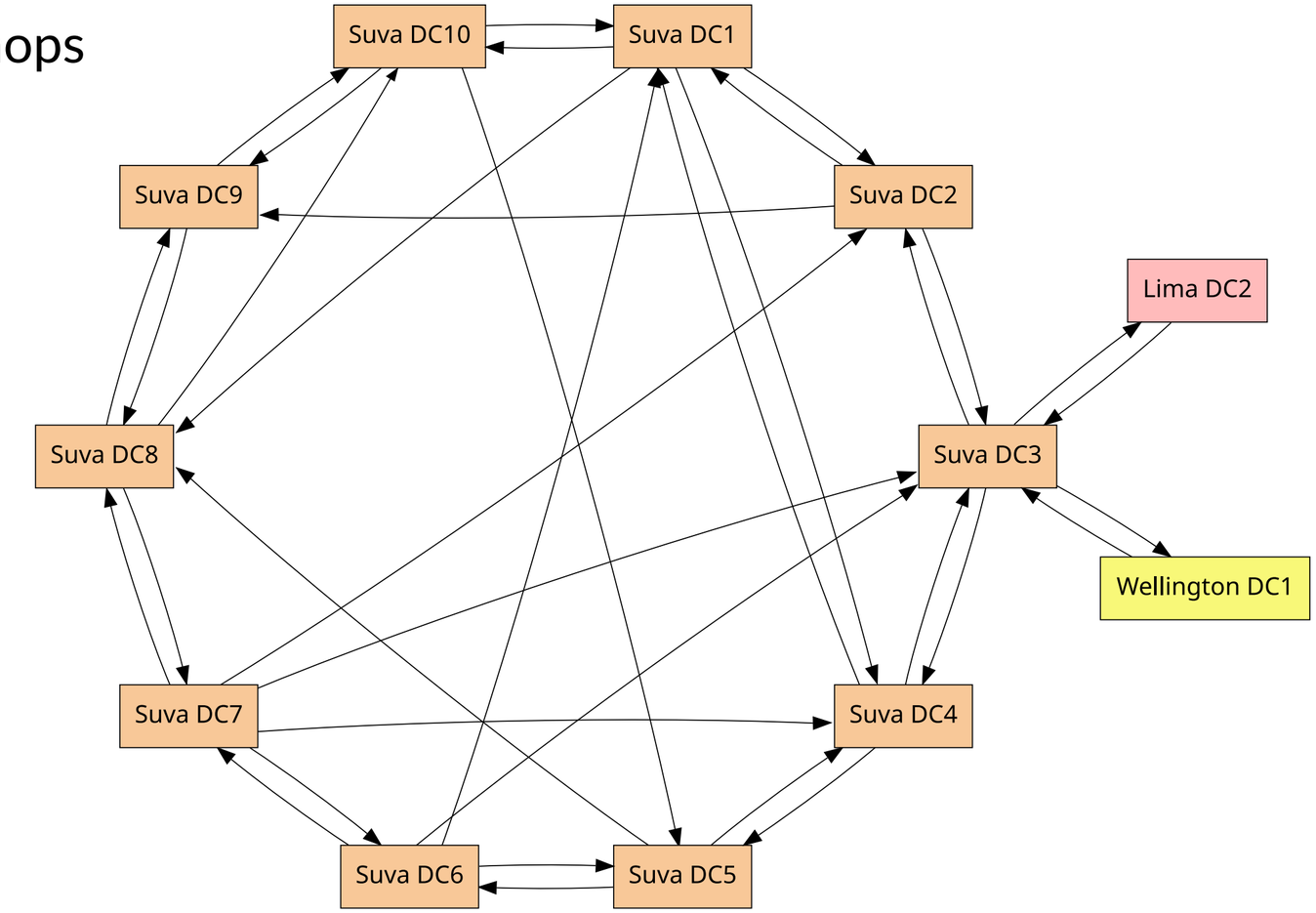



Suva DC2
Lima DC2
Suva DC1
Suva DC3
Suva DC5
Wellington DC1
Suva DC4

With more DCs, add random cross-links



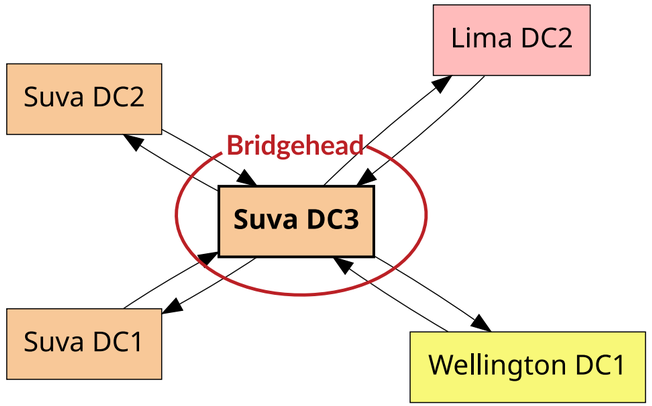
Aiming for a probable maximum of 3 hops



*Inter*site is based on Dijkstra's Algorithm and Kruskal's Algorithm

*Intra*site is the double ring + increasing random links

Some DCs are nominated as intersite bridgeheads



Some DCs are InterSite Topology Generators (ISTGs)

Samba's KCC

The old KCC made fully connected graphs
written in C

samba_kcc is written in Python,
run as a subprocess

started in 2011 by Dave Craft
"finished" in 2015 by Garming Sam and me.

follows [MS-ADTS] 6.2.*

KCC mechanics

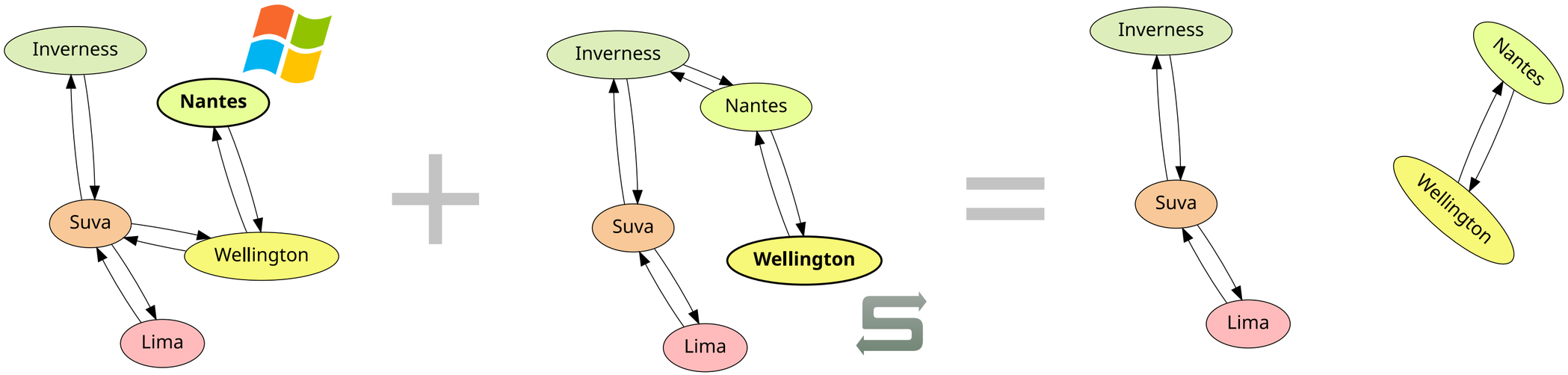
The KCC *thinks* in NTDSConnections,
produces RepsTo and RepsFrom links

Each DC:
- periodically pulls from DCs in its RepsFrom list
- asks its RepsTo partners to pull when necessary

only the RepsTo and RepsFrom really matter.

KCC mismatch

In a mixed domain, if Samba and Windows have different intersite trees,
the network could split.



Samba KCC problems

KCC is
- complex and poorly specified, showing its age
- practically untestable without a large network
  and iterated runs

samba_kcc
  developed in disconected spurts
  by novices
  over many versions of Python
  using buggy Python bindings

working around 2011 Python bugs

```
        # WARNING:
        #
        # There is a very subtle bug here with python
        # and our NDR code.  If you assign directly to
        # a NDR produced struct (e.g. t_repsFrom.ctr.other_info)
        # then a proper python GC reference count is not
        # maintained.
        #
        # To work around this we maintain an internal
        # reference to "dns_name(x)" and "other_info" elements
        # of repsFromToBlob.  This internal reference
        # is hidden within this class but it is why you
        # see statements like this below:
        #
        #   self.__dict__['ndr_blob'].ctr.other_info = \
        #        self.__dict__['other_info'] = drsblobs.repsFromTo1OtherInfo()
        #
        # That would appear to be a redundant assignment but
        # it is necessary to hold a proper python GC reference
        # count.
        if ndr_blob is None:
            self.__dict__['ndr_blob'] = drsblobs.repsFromToBlob()
            self.__dict__['ndr_blob'].version = 0x1
            self.__dict__['dns_name1'] = None
            self.__dict__['dns_name2'] = None

            self.__dict__['ndr_blob'].ctr.other_info = \
                self.__dict__['other_info'] = drsblobs.repsFromTo1OtherInfo()

        else:
            self.__dict__['ndr_blob'] = ndr_blob
            self.__dict__['other_info'] = ndr_blob.ctr.other_info

            if ndr_blob.version == 0x1:
                self.__dict__['dns_name1'] = ndr_blob.ctr.other_info.dns_name
                self.__dict__['dns_name2'] = None
            else:
                self.__dict__['dns_name1'] = ndr_blob.ctr.other_info.dns_name1
                self.__dict__['dns_name2'] = ndr_blob.ctr.other_info.dns_name2

    def __str__(self):
```

Things samba_kcc does not do

- correctly check the liveness of links
- handle SMTP transport

Things samba_kcc does do that are absolutely pointless

- calculate replication schedules that are unused
- calculate all kinds of unused flags

```
NTDSCONN_OPT_TWOWAY_SYNC
DRSUAPI_DRS_DISABLE_AUTO_SYNC
DRSUAPI_DRS_DISABLE_PERIODIC_SYNC
NTDSCONN_OPT_DISABLE_INTERSITE_COMPRESSION
DRSUAPI_DRS_USE_COMPRESSION
NTDSSETTINGS_OPT_IS_TOPL_DETECT_STALE_DISABLED
NTDSCONN_OPT_OVERRIDE_NOTIFY_DEFAULT
NTDSSITELINK_OPT_TWOWAY_SYNC
DS_NTDSSETTINGS_OPT_IS_RAND_BH_SELECTION_DISABLED
```

Unexpected things samba_kcc does

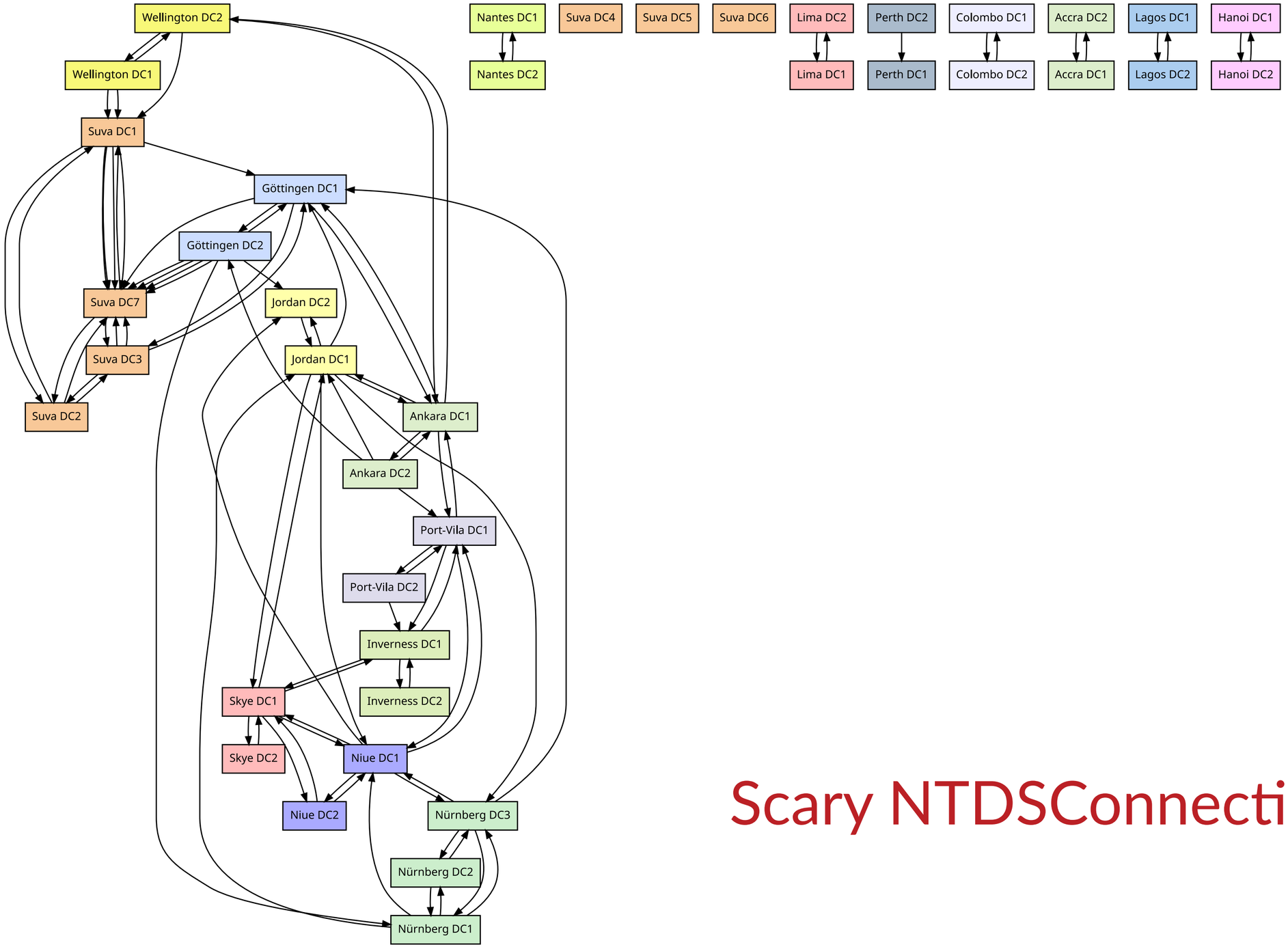
- lots of multi-coloured debug messages, not controlled by **-d**
- lots of self-testing (graph verification)
- writes .dot graph files (like `samba-tool visualize reps --dot`)
- loads fake domains via LDIF for testing
- leaves extra repsTo and repsFroms lying around

samba_kcc

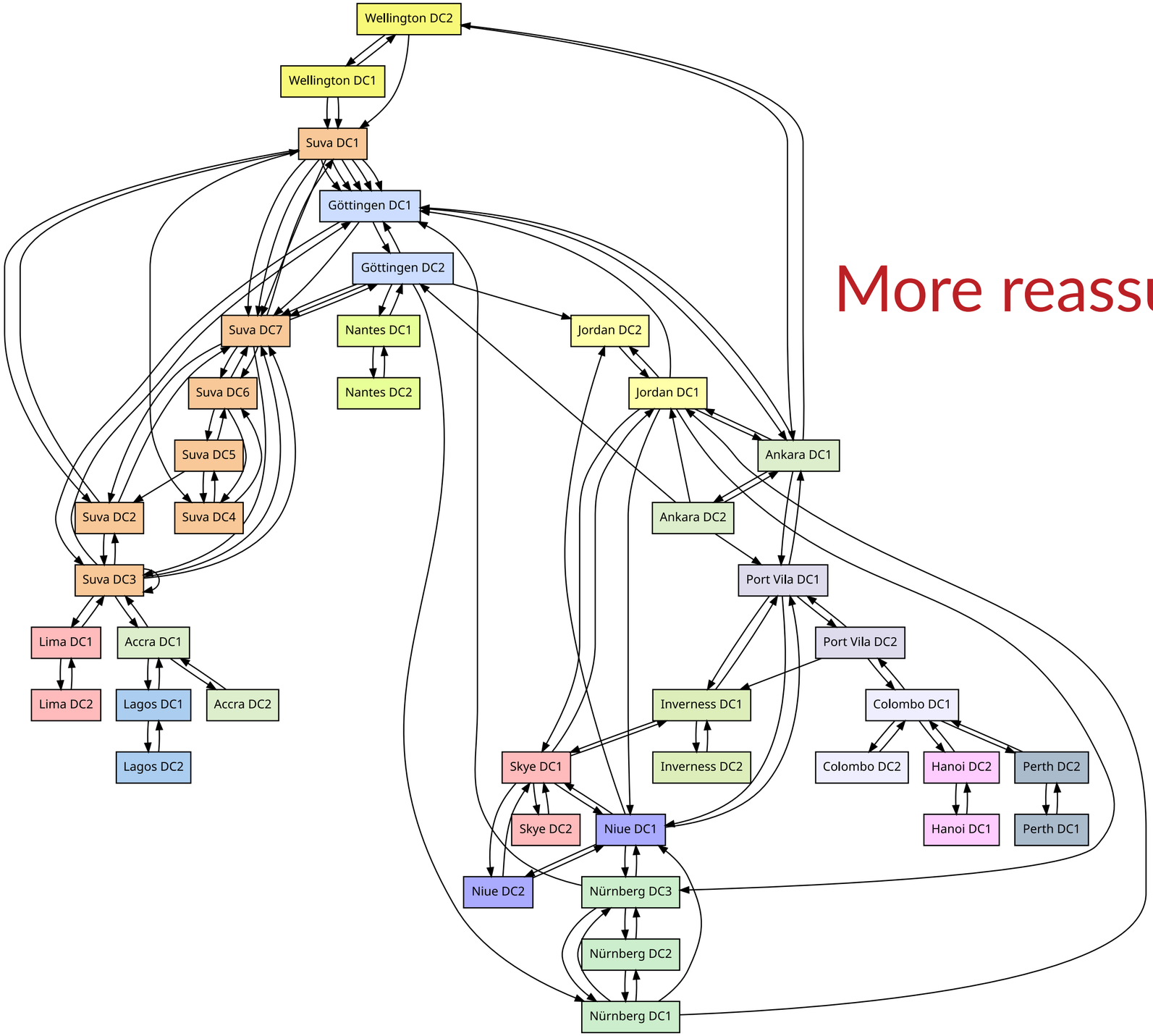
- could be 50% smaller with no loss

- or 90% smaller and work *better*

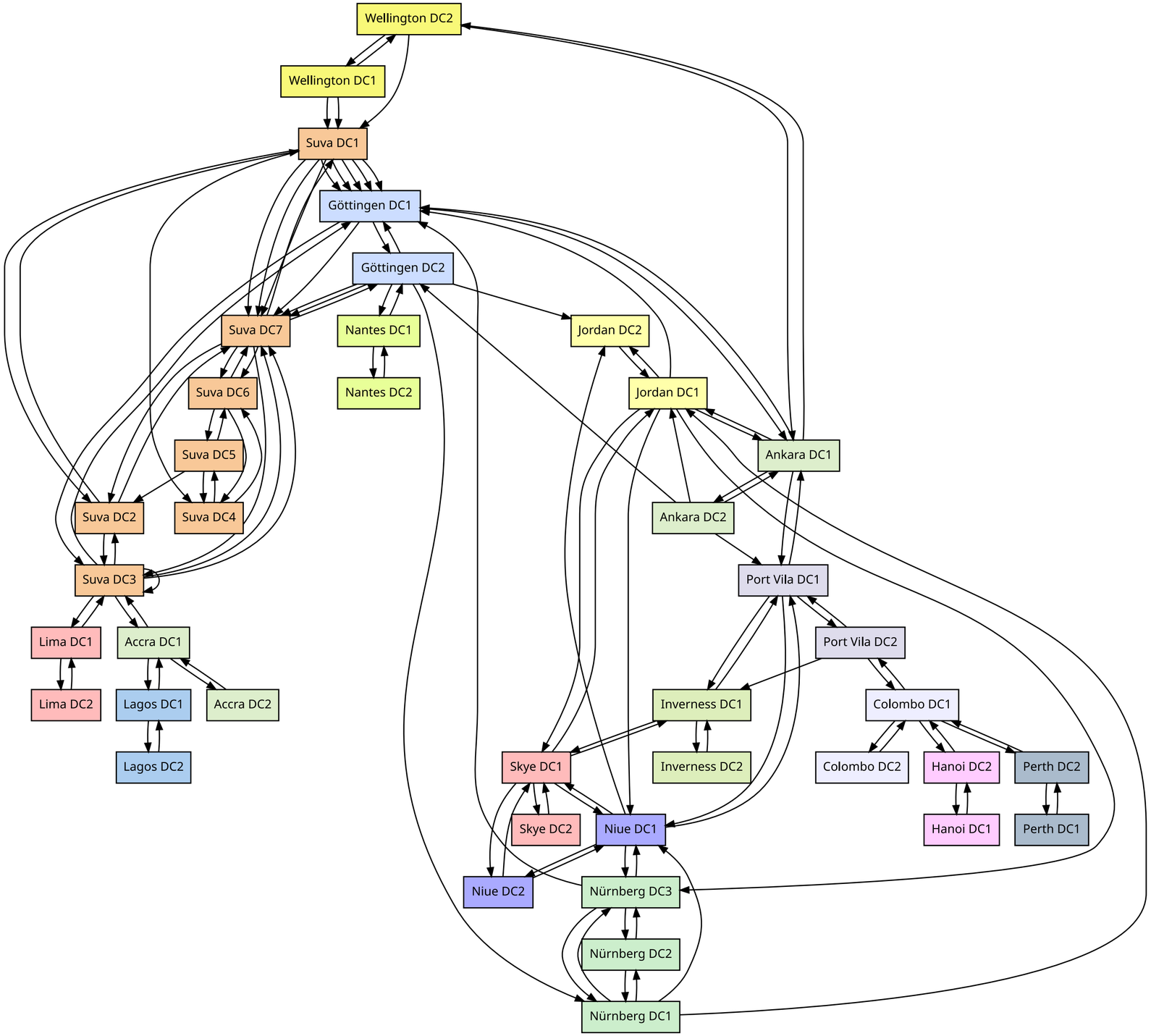
it could carefully diverge from Windows topology, in interoperable ways (we know accidental divergence works OK).



Scary NTDSConnection graph



More reassuring repsFrom graph


Wellington DC2
Wellington DC1
Suva DC1
Göttingen DC1
Göttingen DC2
Suva DC7
Nantes DC1
Jordan DC2
Nantes DC2
Jordan DC1
Suva DC6
Suva DC5
Ankara DC1
Suva DC2
Suva DC4
Ankara DC2
Suva DC3
Port Vila DC1
Lima DC1
Accra DC1
Port Vila DC2
Lima DC2
Lagos DC1
Accra DC2
Inverness DC1
Colombo DC1
Lagos DC2
Skye DC1
Inverness DC2
Colombo DC2
Hanoi DC2
Perth DC2
Skye DC2
Niue DC1
Hanoi DC1
Perth DC1
Niue DC2
Nürnberg DC3
Nürnberg DC2
Nürnberg DC1

Questions?