Redundancy Manual

1. Configuration

- User manual

Redundancy system is just for safety of some features. So it needs 2 independent equipment to run system.

The simple logic of redundancy system is as follows.

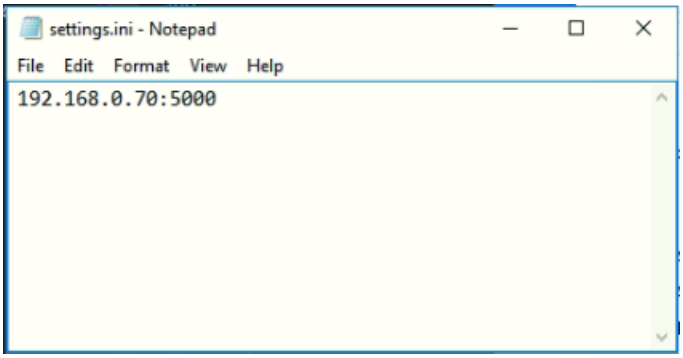
1. Machine A and B run concurrently.
2. At least but at most 1 machine is a master. Master machine is the main machine running the system completely.
3. Two machines should communicate each other to detect the failure of mate machine.
4. If master machines fails to run (slave detects it as no keepalive message from the master for a period of time), the slave machine becomes a master and run complete system.

The logic is simple but there are several acute problems to implement it.

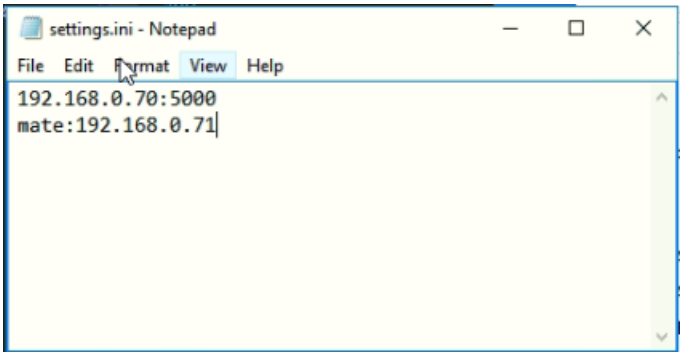
“displaymanager.exe” could work as standalone or redundancy machine.

In settings.ini, there are one/two lines.

If you want to run as a standalone Cpp App, then please remove the second line which is led by “mate:”.



If you want to run as a redundancy mode, then make sure the second line to be “mate:IP”.



IP address should be the address of mate computer and the two machines could be connected directly.

It is allowed to use gateway between machine A and B, but for the sake of safety, it is not recommended.

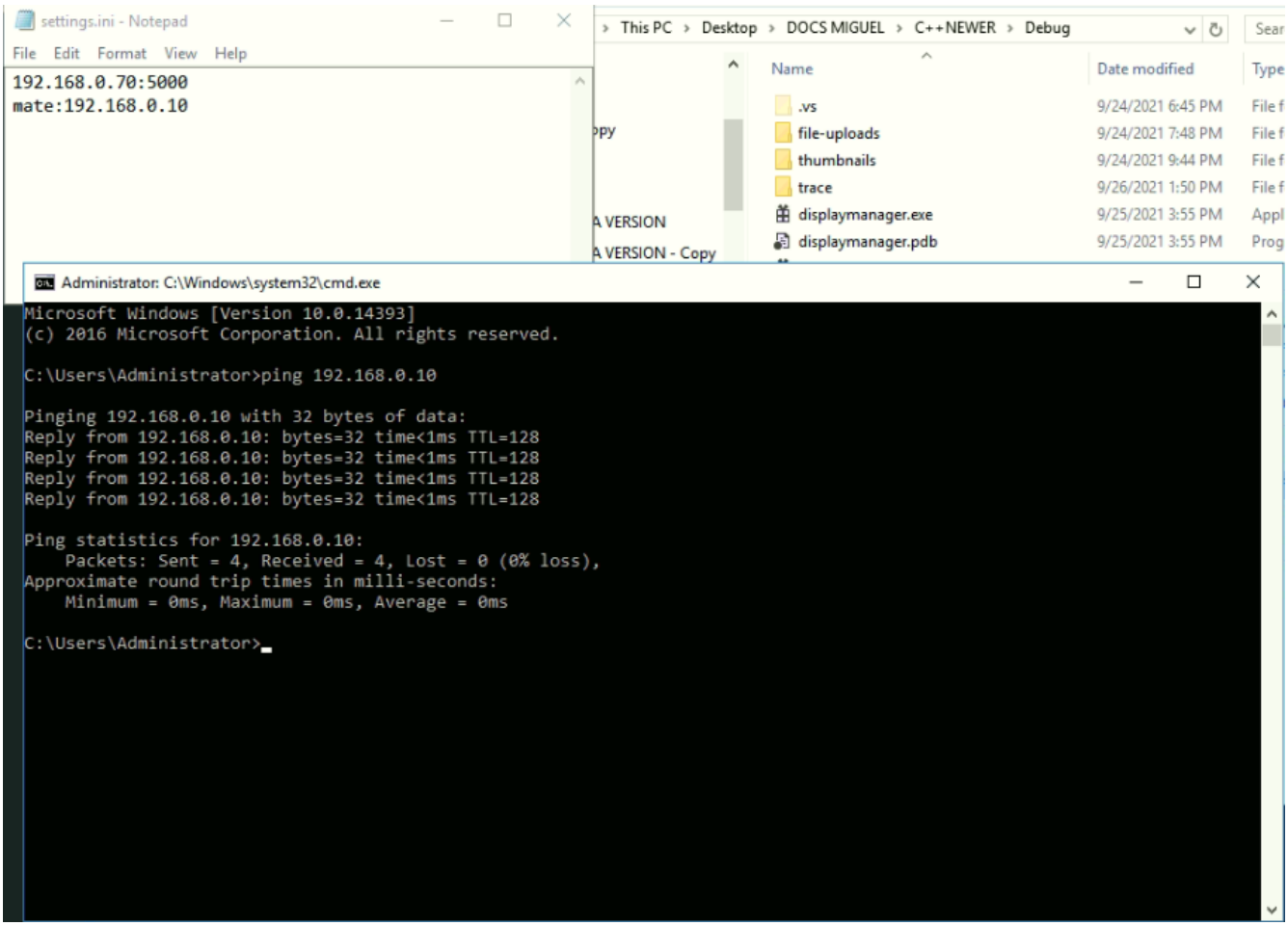
Anyhow, machine A should be ready to ping to machine B IP.

- Playing presentation under redundancy mode

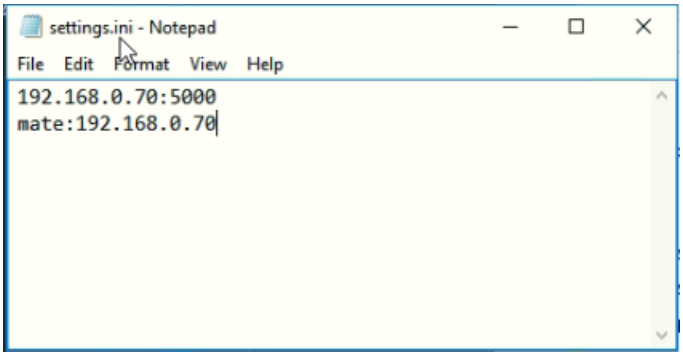
Let’s assume that machine A(192.168.0.70) and machine B(192.168.0.10) are running as redundancy mode.

Then the configuration of each machine should be as follows.

Machine A(192.168.0.70):



Machine B(192.168.0.10):



First run “displaymanager.exe” at machine A.

After 6 seconds, run at machine B.

Then machine A becomes a master for it receives no keepalive message from B for the first 6 seconds.

Machine B becomes a slave for it receives keepalive message from running Cpp App at A.

Both machines send keepalive message every 2 seconds to mate machine.

Just continue playing for some period.

Machine A is a master so that it sends its all presentation data to a slave machine B every 30 seconds.

Machine B receives all the presentation data and synchronize it with its presentation data.

If some errors occur in machine A (for example, powered off, or system went to an error), then machine B will not receive keepalive message from machine A since A failed.

After 6 seconds, B determines that A failed to run, so B becomes a master and plays presentation by synchronized presentation data.

Here are 3 period concepts.

2 seconds: keepalive message period. For master and slave.

30 seconds: delivery of presentation data to a slave. For master.

6 seconds: detection of failure of a master and determination of being a master. For slave.

In the next section, you could find the code associated.

2. Project code

Mate communication instance is created in function CreateMateCommInstance()

If settings.ini does not contain “mate:…” line, then it fails and Cpp App runs as a standalone.

The period of sending a keepalive message is in CMateComm.cpp.

```
#define MATE_COMM_MAX_DELAY		2000 // 2000 milliseconds = 2 seconds
```

The period for determination of a master is also there.

```
#define MATE_NO_MATE_DELAY		6000 // 6000 milliseconds = 6 seconds.
```

UDP port for mate communication is 0x5564 as below.

```
#define MATE_COMM_PORT		0x5563 // 0x5563 = 21859
```

And synchronization of presentation data is in displaymanager.cpp.

For a master, it should send presentation data to a slave every 30 seconds.

It is written in function MainThread().

```
if (curTime > mateSyncTime + 30)
```

Here 30 means 30 seconds. If you want to change the period, then change this value. Note that this value could not be enough small. It is recommended to be greater than 20 seconds for safety.