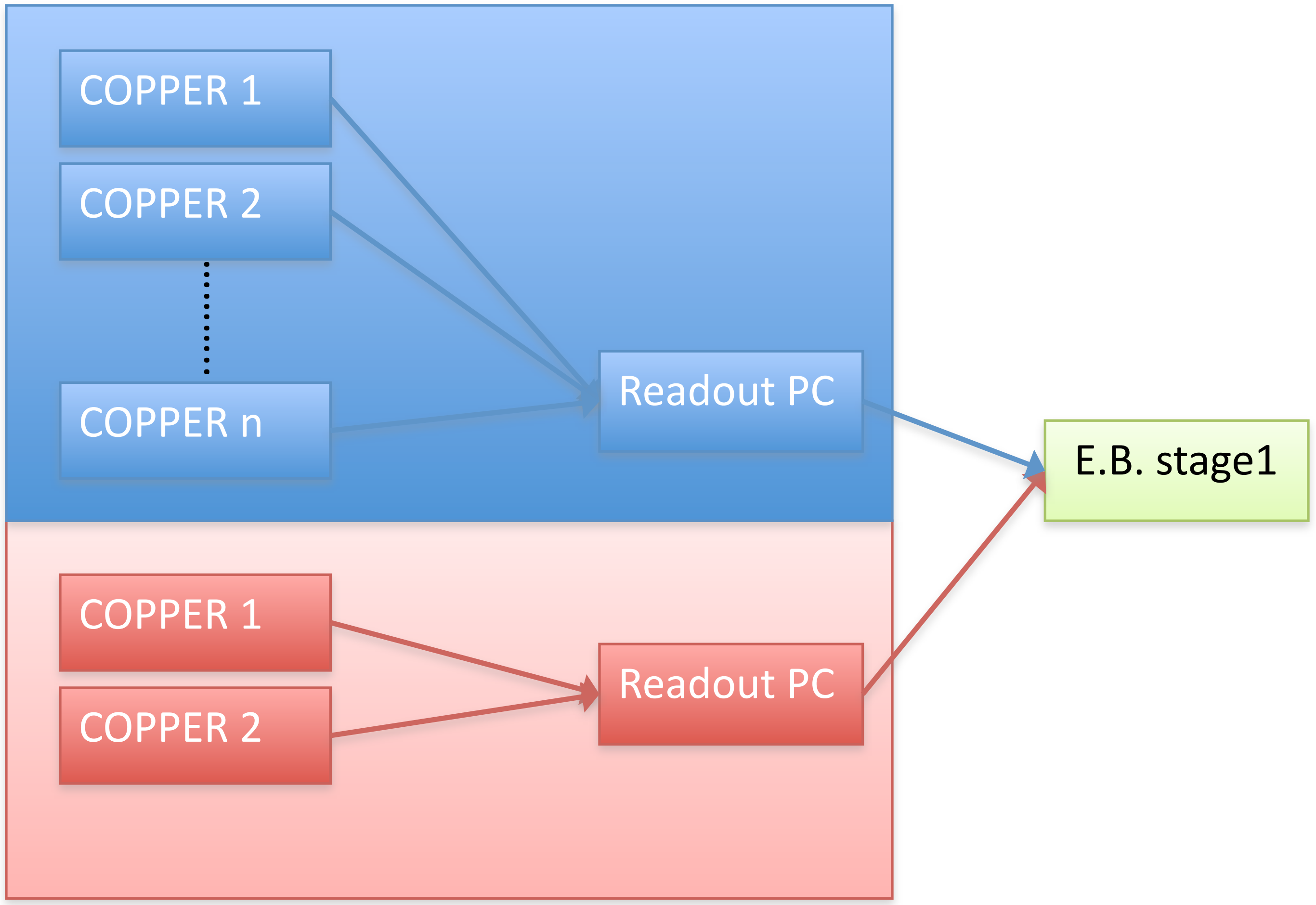
E.B. for SuperBelle

T.Higuchi & yamagata

Current data flow

- Connection making: from upstream to downstream



Problem

- When something bad happen in downstream, all of upstream must be restarted by exp-shift.
  - # of upstream > # of downstream
  - It takes long long time, even in current DAQ
  - Will be more longer in SuperBelle.

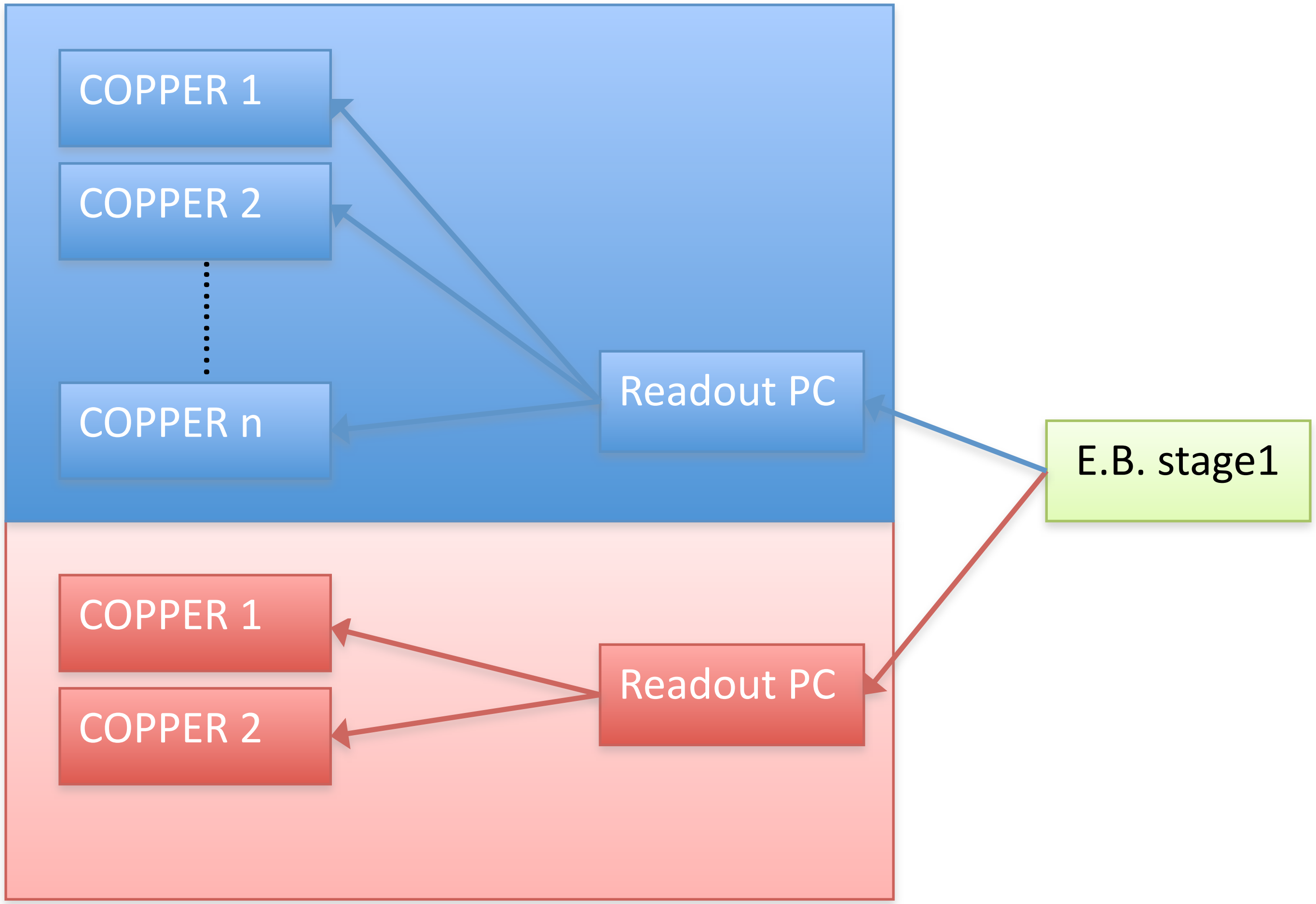
- All components rely on the "status" of NSM
  - But sometimes components are not fully ready even if it says "READY".

Reduce # of restart components

- Connection from downstream to upstream may be effective.
  - If one upstream which has been connected just now found in any problem, all downstream nodes are restarted.
  - Even in the case, the # of restart nodes == # of downstream < # of upstream

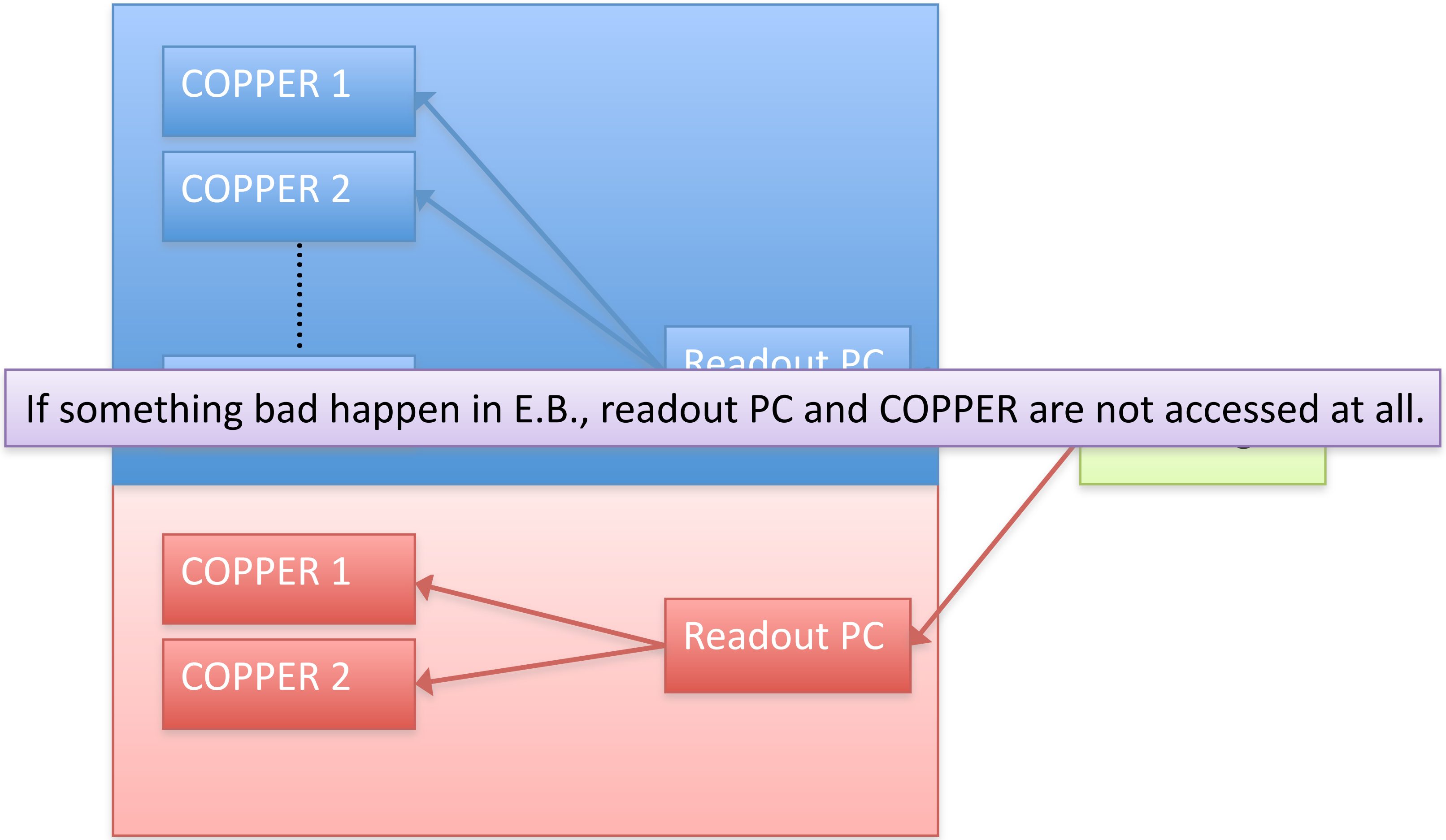
Connection from down to up

Readout PC doesn't access COPPERs until it is accessed from E.B.



Connection from down to up

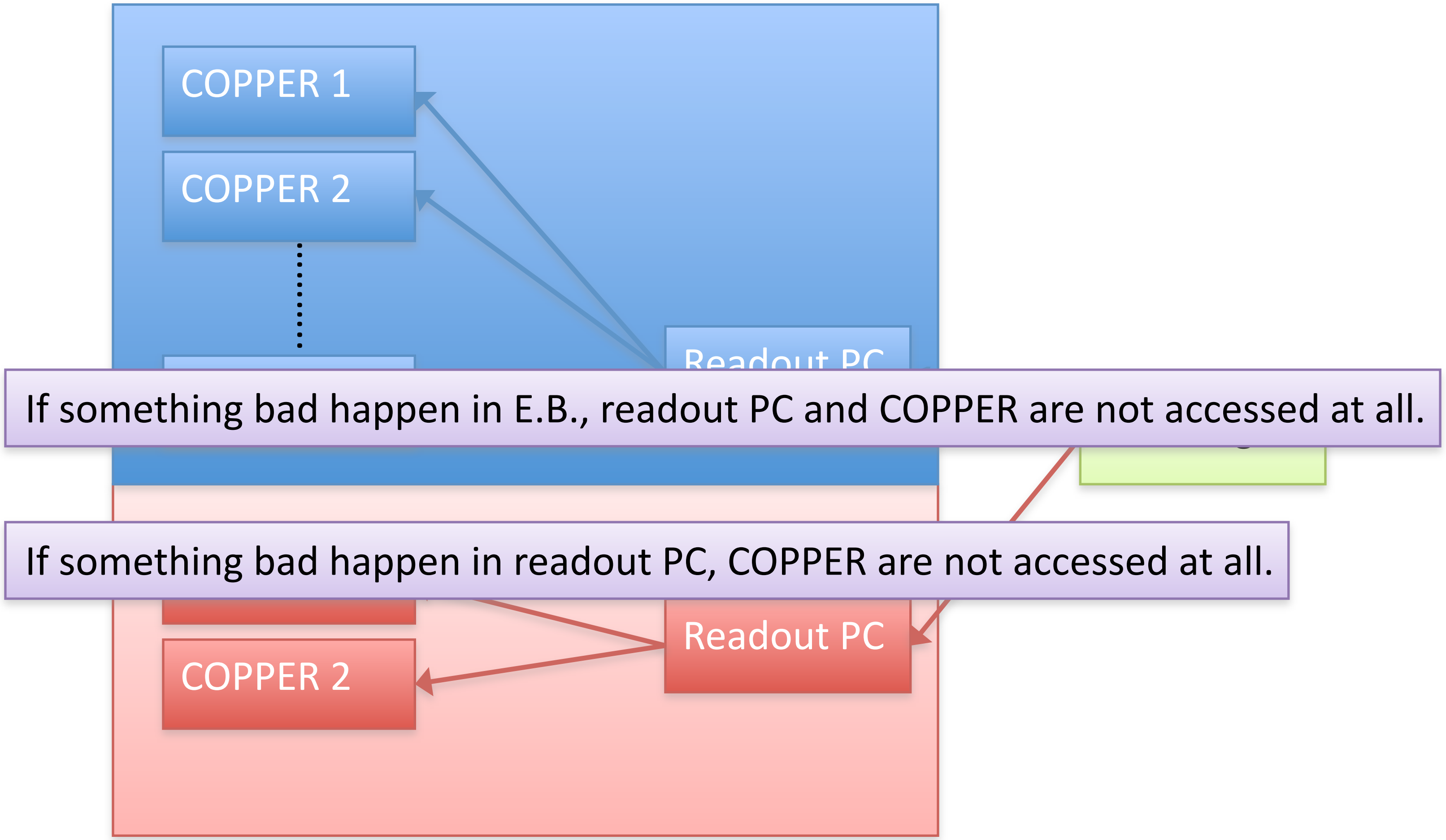
Readout PC doesn't access COPPERs until it is accessed from E.B.



If something bad happen in E.B., readout PC and COPPER are not accessed at all.

Connection from down to up

Readout PC doesn't access COPPERs until it is accessed from E.B.



If something bad happen in E.B., readout PC and COPPER are not accessed at all.

If something bad happen in readout PC, COPPER are not accessed at all.

Difficulty about status

- The reason of the RUN start failure is very, very various.
  - Dead socket connection
  - A wreck of Shared Memory or Semaphore
- Pre-defined STATUS message is too short to tell the exact way to the recovery way.
- We are developing stateless system event building and COPPER readout

What is the stateless?

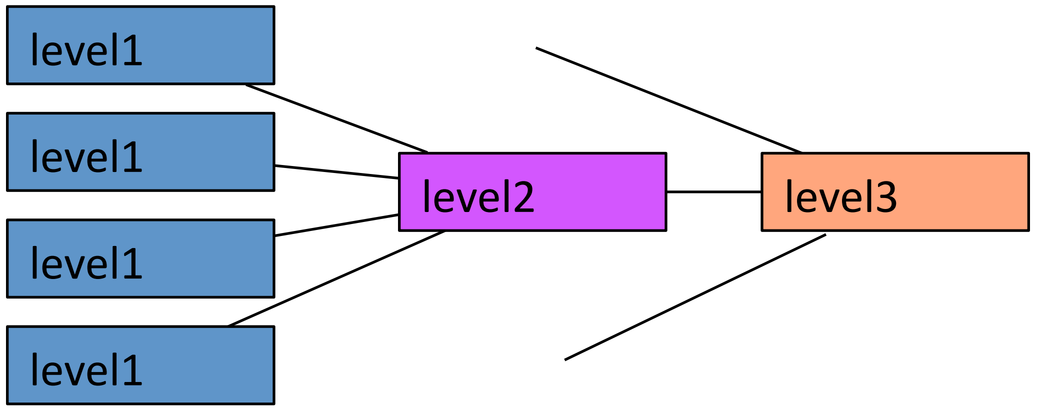
- If RUN is not started, no process is running.
- All processes of E.B. are,
  - kicked from inetd at the RUN start
  - killed by connection close at the RUN end

But we want to know all COPPER status before the RUN start!

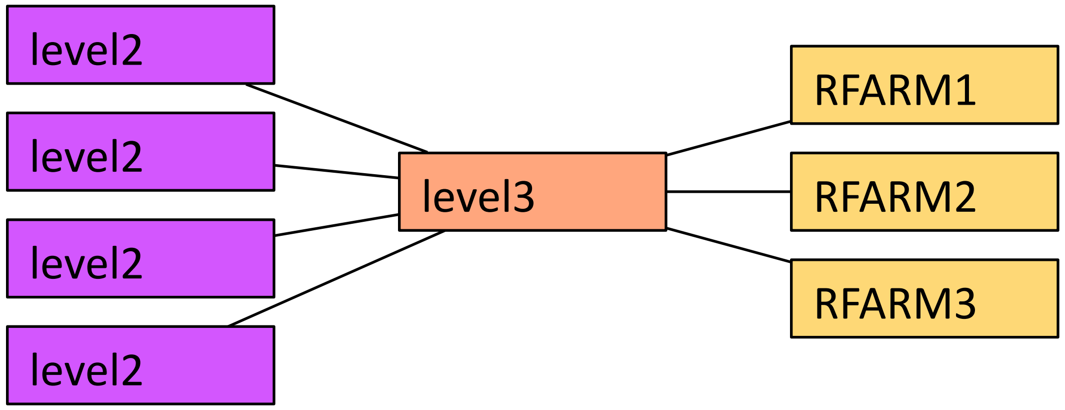
- Service Discovery such as
  - SunRPC
  - ZEROconf
    - Bonjour
    - Avahi
- But we have rwho/rwhod
  - We can detect host status (up/down)
  - Other status can be adopted via utmp

Two types for E.B. node

- Multiple input,
  Single output
  - traditional E1,2,3



- Multiple input,
  Multiple output



Why?

- Current Scheme
  - E.B. software locates in EFARM
  - # of EFARM == # of RFARM
    - Each of RFARM has their own EFARM
- In the case of Super Belle
  - # of RFARM will be very large
  - # of PC in EFARm will be awfully large
  - large # of EFARM makes it hard to manage
- We will build only one EFARM
  - so multiple in/out component is needed.

Inetd childlen can't handle multiple downstream!

- Inetd kicks the child program for each accept of the connection.
- So each of the process has only one downstream socket.
- But E.B. must distribute data to multiple RFARM.
  - File descriptor passing to other process

File descriptor passing

man –s 2 recv says,

……

Open file descriptors are now passed as ancillary data for AF_UNIX domain sockets, with cmsg_level set to SOL_SOCKET and cmsg_type set to SCM_RIGHTS.

……

File descriptor passing

man –s 2 recv says,

……

Open file descriptors are now passed as ancillary data for AF_UNIX domain sockets, with cmsg_level set to SOL_SOCKET and cmsg_type set to SCM_RIGHTS.

……

(1)

Connection from RFARM1

$\longrightarrow$

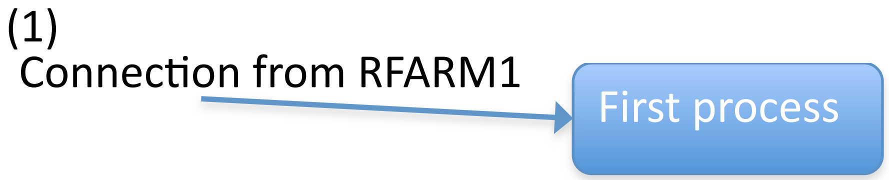
File descriptor passing

man –s 2 recv says,

……

> Open file descriptors are now passed as ancillary data for AF_UNIX domain sockets, with cmsg_level set to SOL_SOCKET and cmsg_type set to SCM_RIGHTS.

……



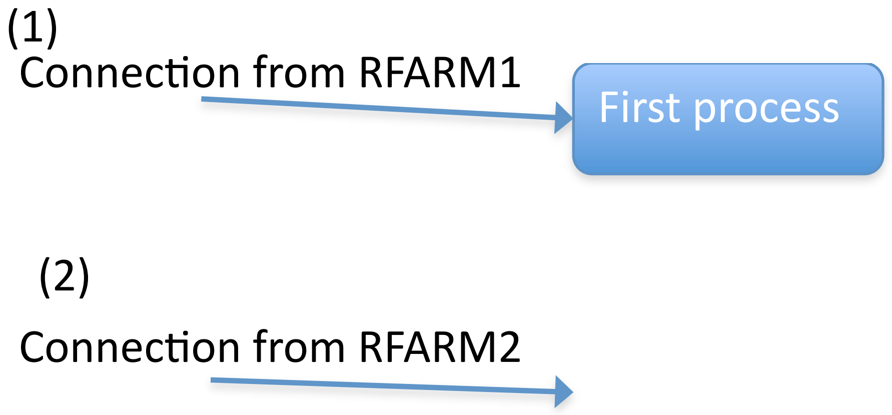
File descriptor passing

man –s 2 recv says,

……

Open file descriptors are now passed as ancillary data for AF_UNIX domain sockets, with cmsg_level set to SOL_SOCKET and cmsg_type set to SCM_RIGHTS.

……



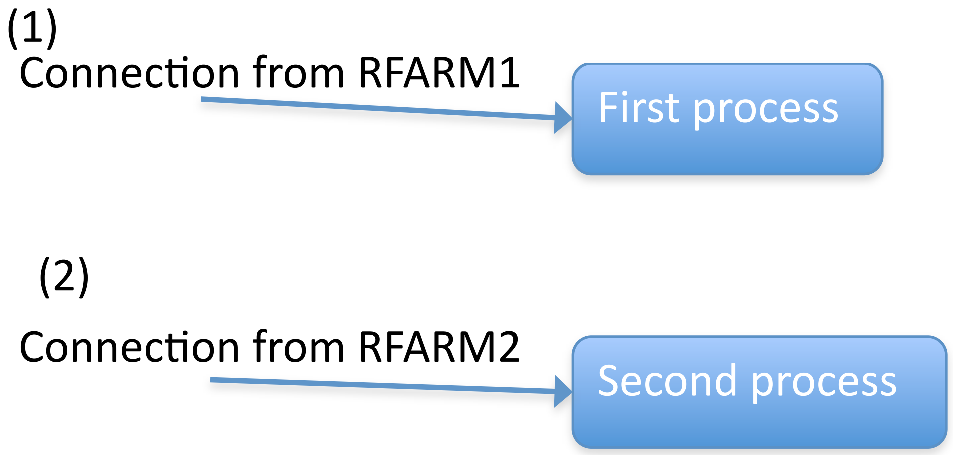
File descriptor passing

man –s 2 recv says,

……

> Open file descriptors are now passed as ancillary data for AF_UNIX domain sockets, with cmsg_level set to SOL_SOCKET and cmsg_type set to SCM_RIGHTS.

……



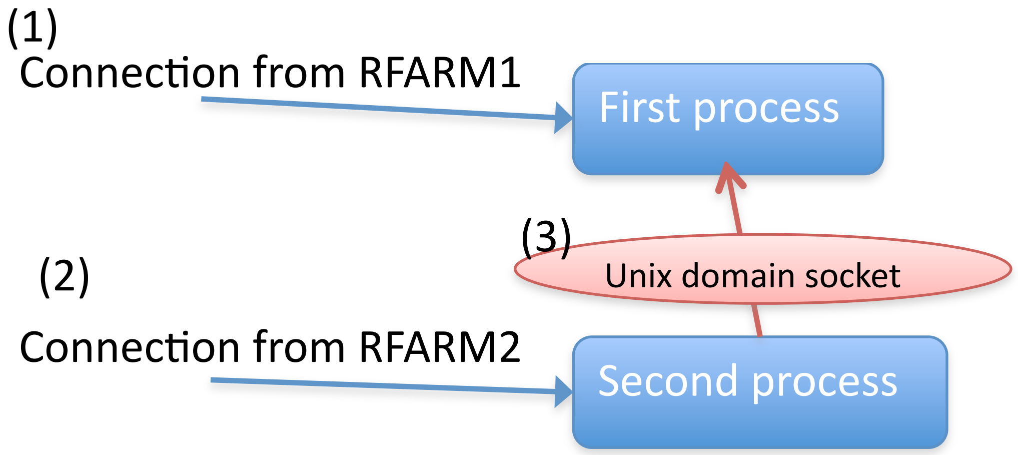
File descriptor passing

man –s 2 recv says,

……

Open file descriptors are now passed as ancillary data for AF_UNIX domain sockets, with cmsg_level set to SOL_SOCKET and cmsg_type set to SCM_RIGHTS.

……



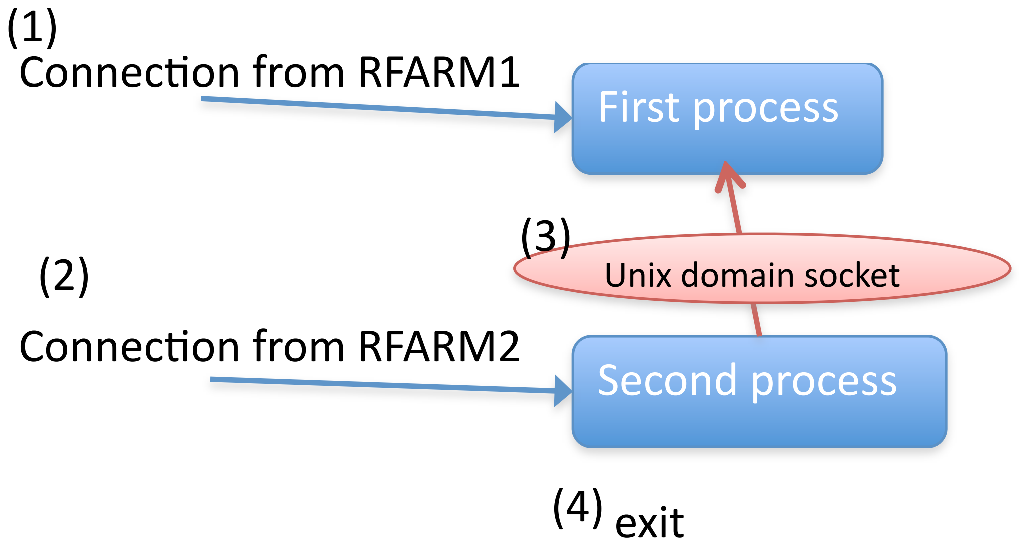
File descriptor passing

man –s 2 recv says,

……

Open file descriptors are now passed as ancillary data for AF_UNIX domain sockets, with cmsg_level set to SOL_SOCKET and cmsg_type set to SCM_RIGHTS.

……



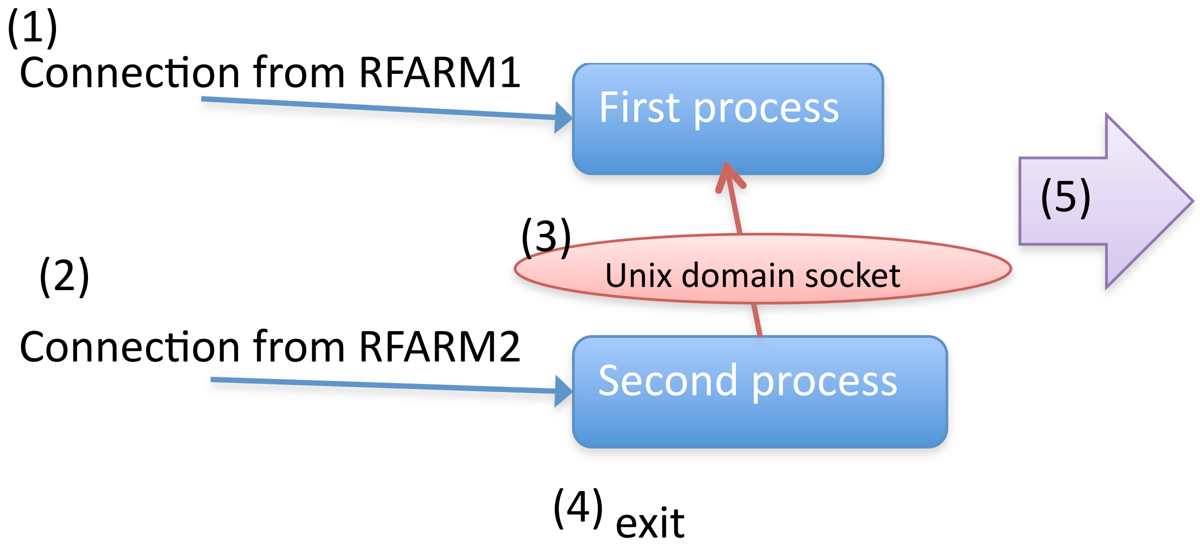
File descriptor passing

man –s 2 recv says,

……

> Open file descriptors are now passed as ancillary data for AF_UNIX domain sockets, with cmsg_level set to SOL_SOCKET and cmsg_type set to SCM_RIGHTS.

……



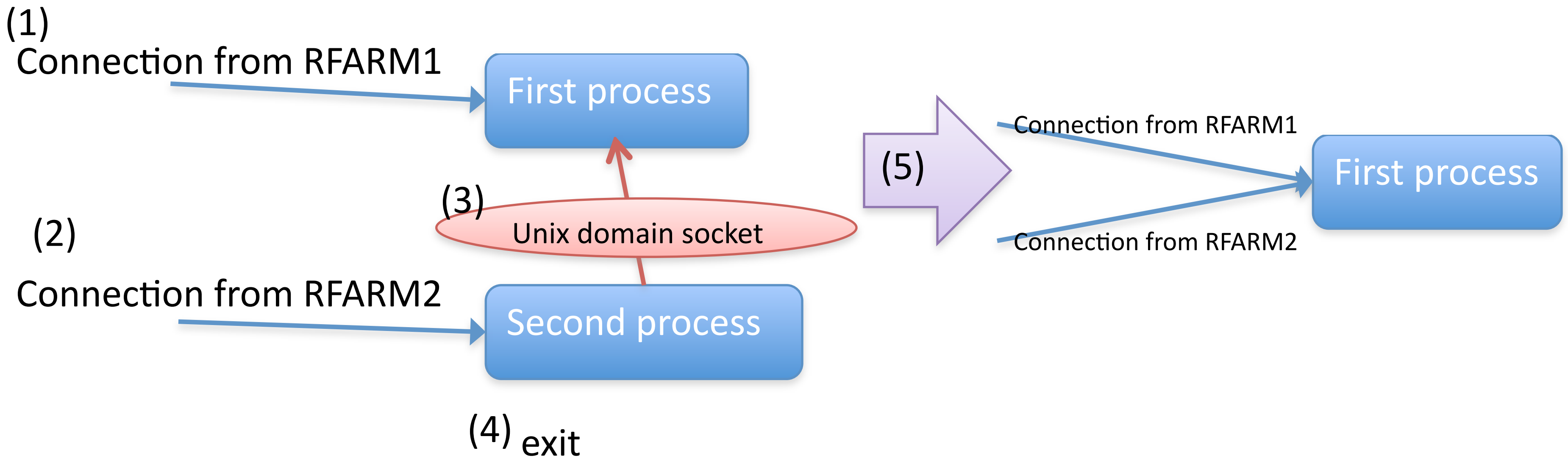
File descriptor passing

man –s 2 recv says,

……

Open file descriptors are now passed as ancillary data for AF_UNIX domain sockets, with cmsg_level set to SOL_SOCKET and cmsg_type set to SCM_RIGHTS.

……



How about CPU consumption?

- When all the data streams are handled by only one process, can the process achieve sufficient throughput?
- Can really single process sends to multiple machines via 1Gbp links?

How about CPU consumption?

- When all the data streams are handled by only one process, can the process achieve sufficient throughput?
- Can really single process sends to multiple machines via 1Gbp links?

Traditionally, it is NO.

How about CPU consumption?

- When all the data streams are handled by only one process, can the process achieve sufficient throughput?
- Can really single process sends to multiple machines via 1Gbp links?

  Traditionally, it is NO.

  1Gbps link consumes 1GHz CPU power.

How about CPU consumption?

- When all the data streams are handled by only one process, can the process achieve sufficient throughput?
- Can really single process sends to multiple machines via 1Gbp links?

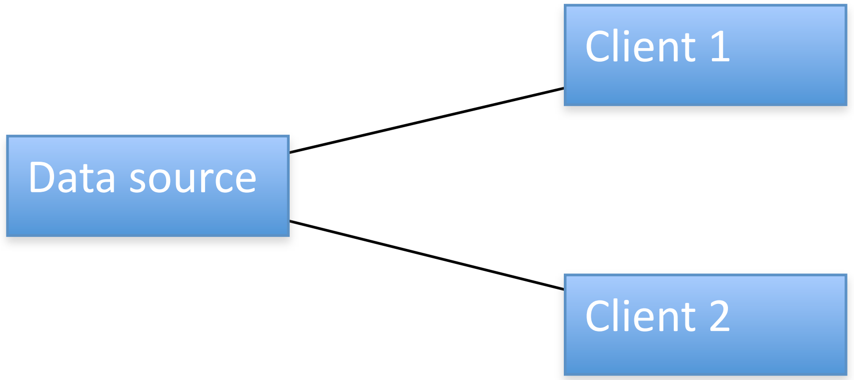
Traditionally, it is NO.

1Gbps link consumes 1GHz CPU power.

1Gbps receiving + sending requires 2GHz.

Test

- Data source
  - Xeon 3.4GHz (C2D Gen.)
  - FSB 1600
- Client 1
  - Xeon 3.0GHz (C2D Gen.)
  - FSB 1333
- Client 2
  - Pentium4 2.6GHz
  - FSB 533
- Connected via e1000, point-to-point



Check the throughput and CPU consumption

Observed speed at client side

#tdiff=time to receive 10M words from server

tdiff=0.356304(sec) 117.717006

tdiff=0.356299(sec) 117.718658

tdiff=0.356540(sec) 117.639087

tdiff=0.356301(sec) 117.717997

tdiff=0.356541(sec) 117.638757

....

117MB/s, fully occupied GbE

CPU consumption

- 7-8% at the server side

```
top - 16:42:52 up 4 days, 22:30,  3 users,  load average: 0.06, 0.11, 0.09
Tasks: 162 total,   1 running, 157 sleeping,   4 stopped,   0 zombie
Cpu(s):  0.0%us,  1.0%sy,  0.0%ni, 97.9%id,  0.0%wa,  0.2%hi,  0.9%si,
0.0%st
Mem:  16632668k total,   714200k used, 15918468k free,   198828k buffers
Swap:        0k total,        0k used,        0k free,   415912k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
17578 nobody    15   0  2668 1532 1456 S    8  0.0   2:06.43 fdpass
    1 root      15   0  2060  660  572 S    0  0.0   0:03.31 init
    2 root      RT  -5     0    0    0 S    0  0.0   0:00.00 migration/0
    3 root      34  19     0    0    0 S    0  0.0   0:00.00 ksoftirqd/0
    4 root      RT  -5     0    0    0 S    0  0.0   0:00.00 watchdog/0
```

CPU consumption

- Client 1 (Xeon 3.0GHz)

```
top - 16:31:02 up  5:04,  3 users,  load average: 0.43, 0.24, 0.13
Tasks:  59 total,   1 running,  58 sleeping,   0 stopped,   0 zombie
Cpu(s):  2.0% us,  2.0% sy,  0.0% ni, 92.5% id,  0.0% wa,  0.0% hi,  3.5% si
Mem:  16439168k total,  1697956k used, 14741212k free,   276380k buffers
Swap:        0k total,        0k used,        0k free,   308216k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 4298 yamagata  15   0 84452  40m  416 S 11.0  0.3   3:30.07 xxx
    1 root      16   0  4772  588  488 S  0.0  0.0   0:00.69 init
```

CPU consumption

- Client 2

```
top - 16:48:46 up  5:23,  2 users,  load average: 0.47, 0.50, 0.42
Tasks:  61 total,   2 running,  55 sleeping,   4 stopped,   0 zombie
Cpu(s):  0.5% us, 10.4% sy,  0.0% ni, 88.1% id,  0.0% wa,  1.0% hi,
0.0% si
Mem:   3114716k total,   276328k used,  2838388k free,    43820k buffers
Swap:  2096472k total,        0k used,  2096472k free,   136880k cached
Change delay from 1.0 to:
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
17179 yamagata  15   0 42724  40m  408 R 20.9  1.3   5:45.34 xxx
17203 yamagata  16   0  2672  968  768 R  1.0  0.0   0:00.14 top
    1 root      16   0  1948  548  472 S  0.0  0.0   0:00.73 init
    2 root      RT   0     0    0    0 S  0.0  0.0   0:00.44 migration/0
    3 root      34  19     0    0    0 S  0.0  0.0   0:00.01 ksoftirqd/0
```

This indicates

- CPU consumer is not sender, but receiver
  - may be because of TCP Segmentation Offload.
- Single process can distribute a few GbE data stream to multiple machines
  - But the bus bandwidth will limit total network bandwidth around 5Gbps from my experience about 10GbE NIC.

So we continue this strategy

- But investigation is still necessary
  - Dependency
    - seems to be depend on FSB speed
    - how about i7?
- More PCs are necessary to test
  - As we have only old PCs, extrapolate is danger

Summary

- Multiple input/output skeleton is ready to test
  - will be usable up to a few Gbps output
  - Must be tested by newer PCs
- In the next step, we will confirm it co-works with stateless neighbor detection based on rwho service.