

E.B. *status*

yamagata

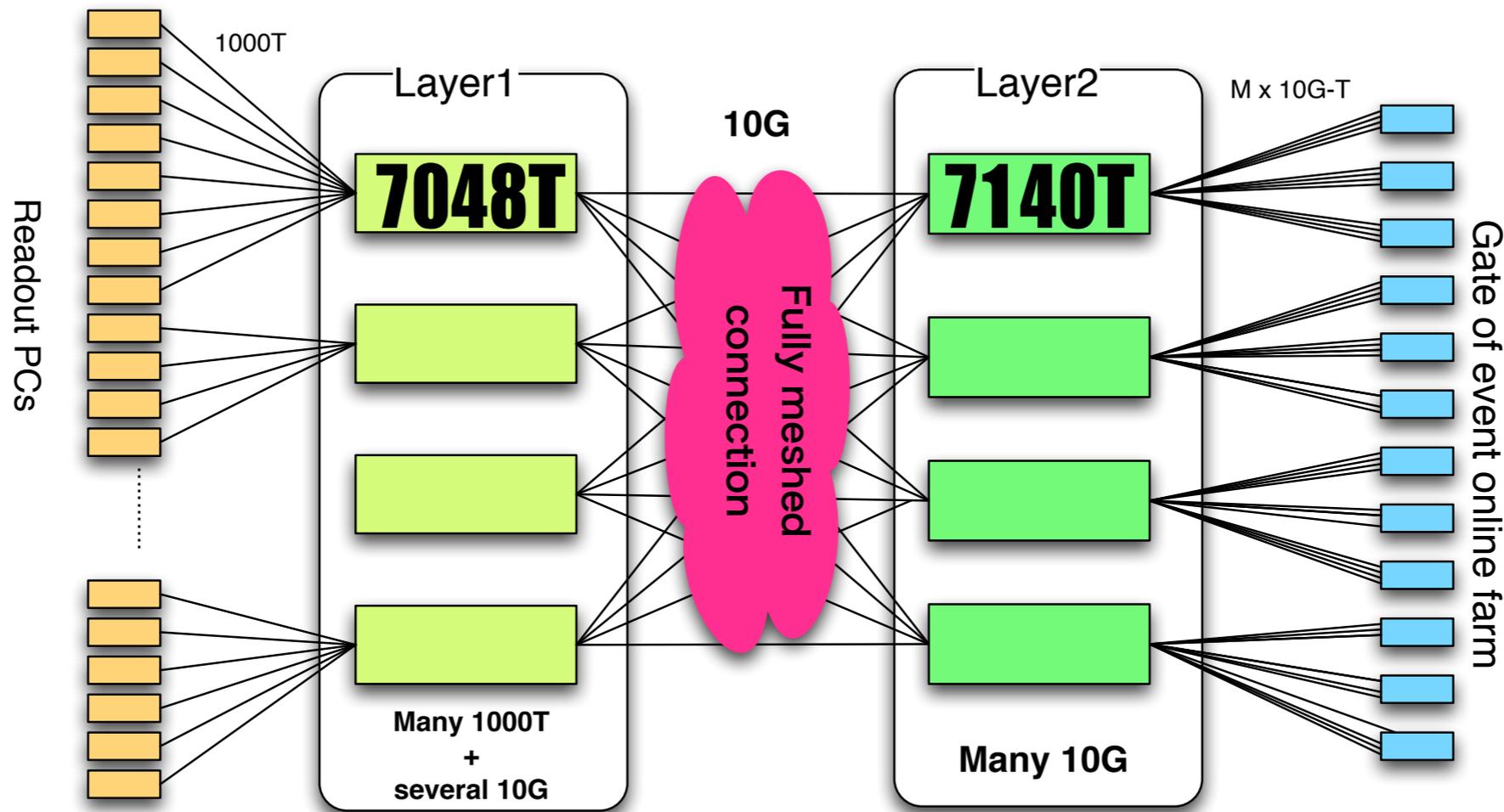
E.B.1 and E.B.2

- E.B.1
 - for all detectors but PXD to HLT farm
- E.B.2
 - for PXD and HLTed data to online storage

E.B.1

- receives data from Readout PCs (ROPC)
 - one ROPC is assigned to one COPPER crate
 - # of ROPCs is about 50
- sends them to online farm
 - # of online farm will starts from 5 to 10.
- consists of 8 network switches and PCs

Current design



- Each ROPC sends data via single GbE
- Layer2 switch and Layer3 are connected by **multiple** 10G

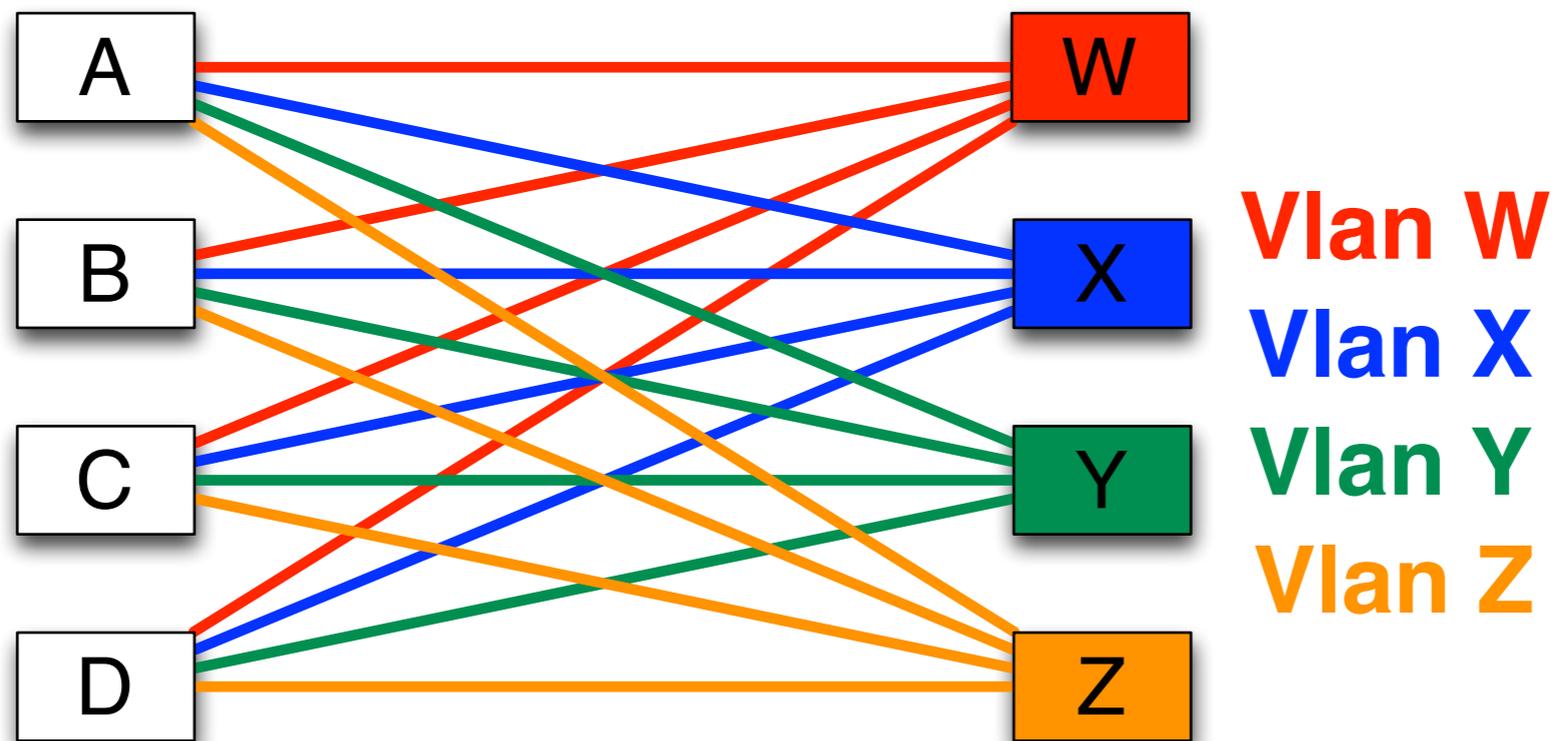
Switches

- 7048T
 - 48 ports of GbE
 - 4 ports of 10GbE
 - deep buffer (768MB per box)
- 7140T or 7148SX
 - 48 ports of 10GbE
 - short buffer (a few MB per
 - not intend to handle GbE



Full mesh requires VLAN separation

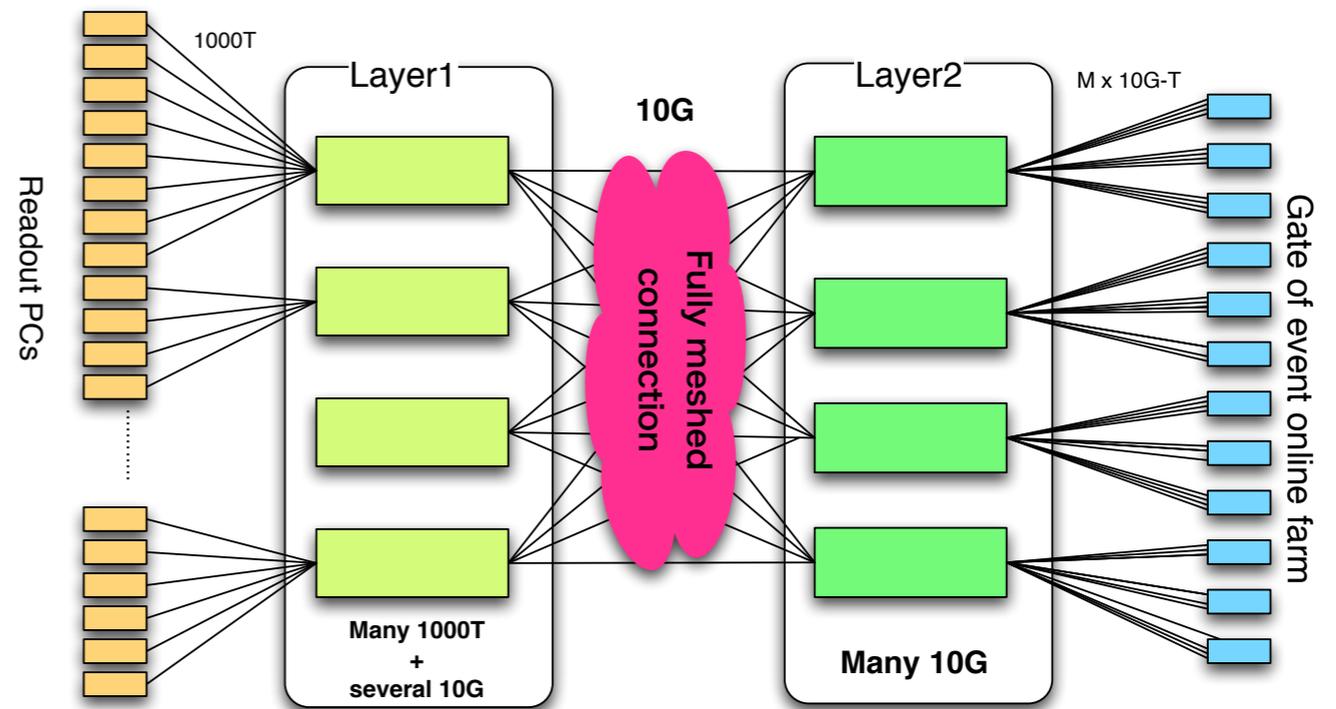
If all links are in identical L2 network,
the network loop problem happen.



ROPC sends packet to switch with VLAN tag.
Linux 8021q driver can do it.

Naming

- “Layer” is not suitable for network equipments. because of OSI layer model. (L1 => phy, L2 => MAC, L3 => IP, ...)
- Call PC at the last stage of EB as “EBPC”, in this slide.



Data packet flow

- COPPER

- ROPC

E.B.1

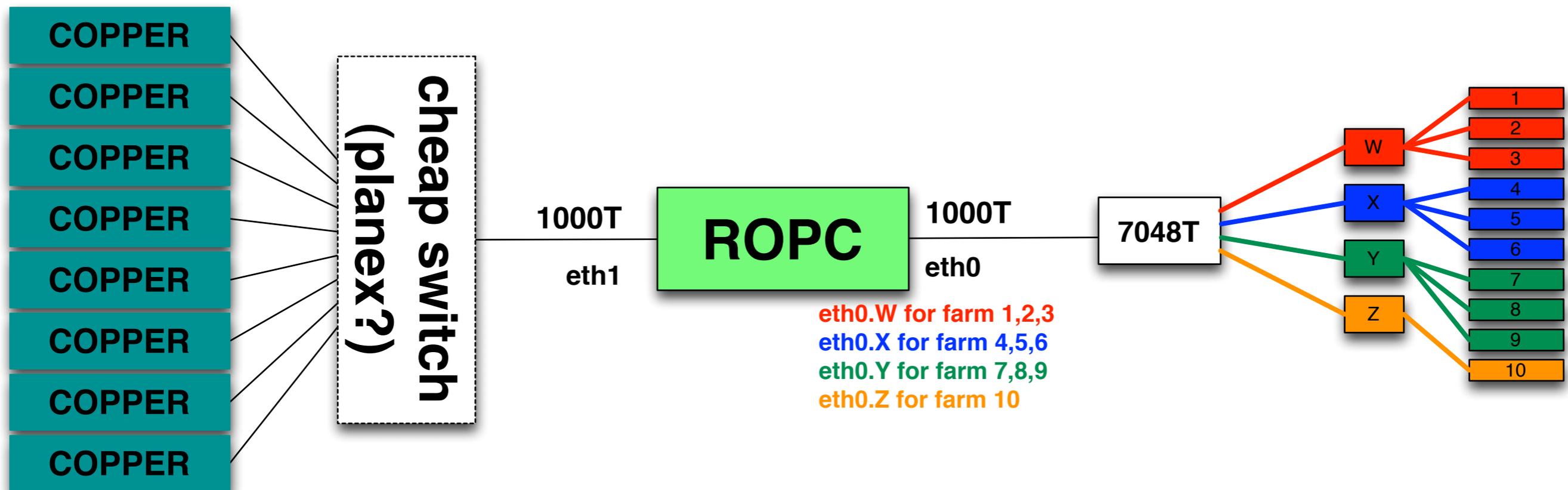
- 7048T

- 7140T or 7148SX

- EBPC

- Online farm for HLT

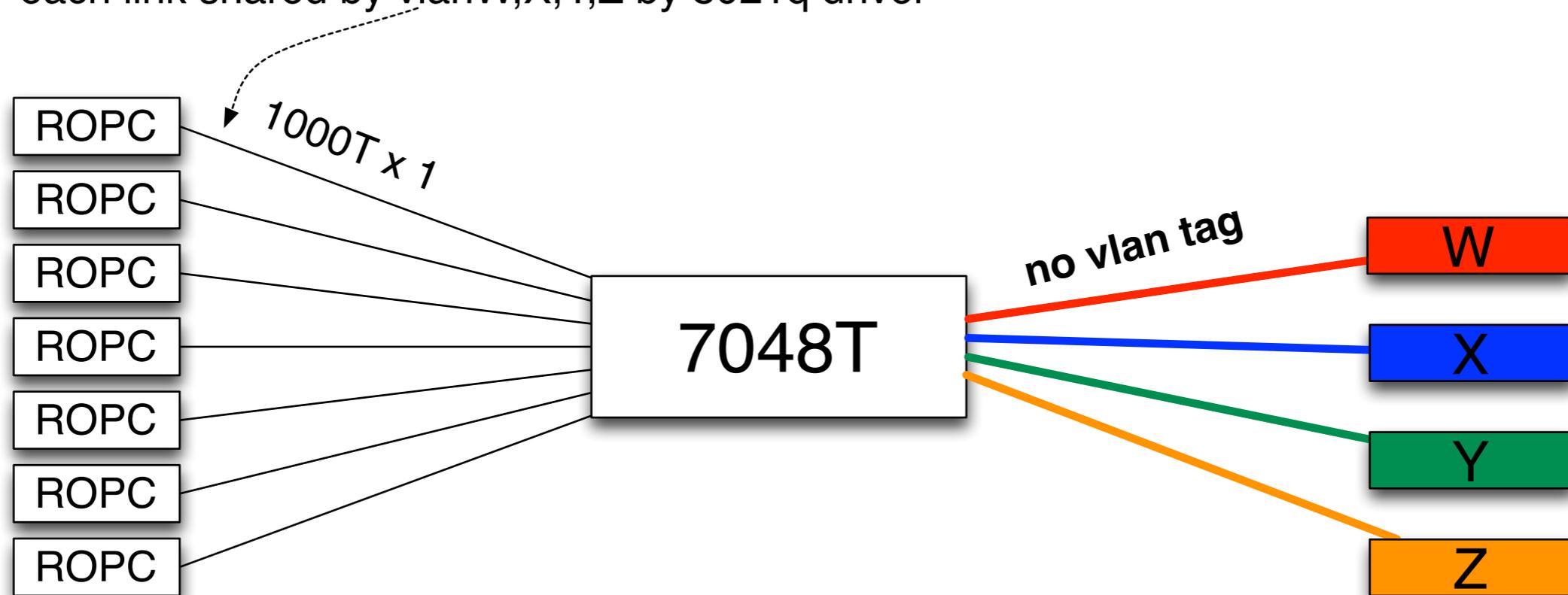
View at ROPC



ROPC accepts connection from all EBPCs.
When all downstream connections become ready,
ROPC starts to connect upstream COPPERs.

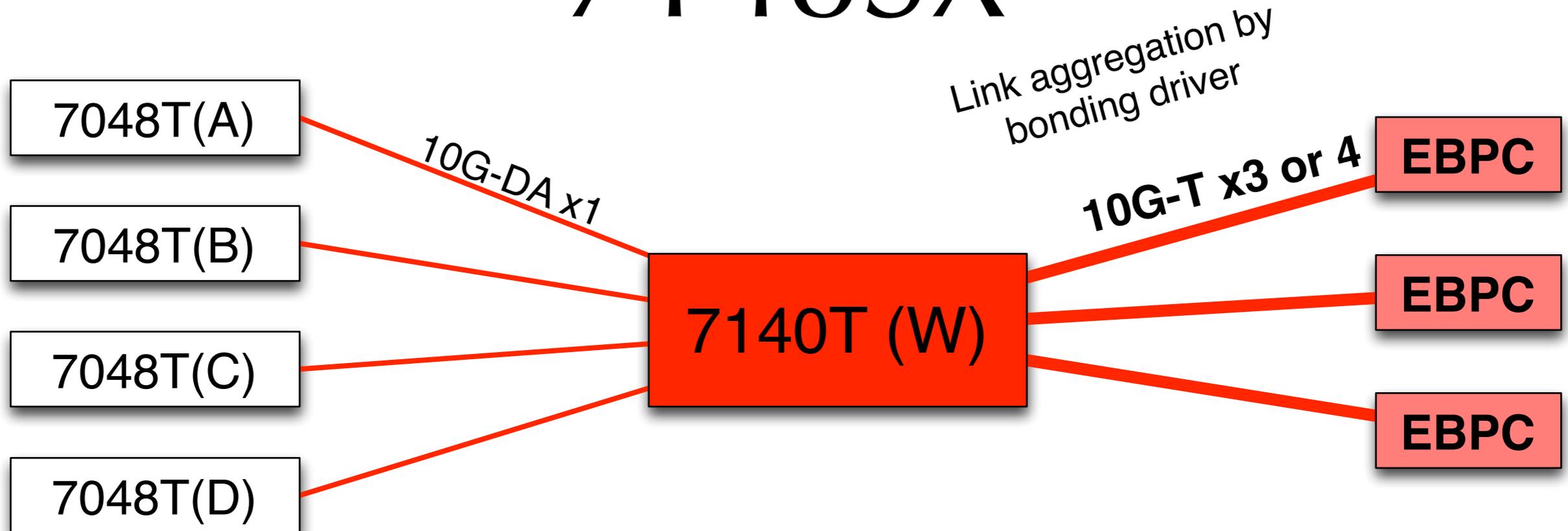
View at 7048T

each link shared by vlanW,X,Y,Z by 8021q driver



7048T has only 4 ports for 10GbE,
so downlink to 7140 will be single link of 10G

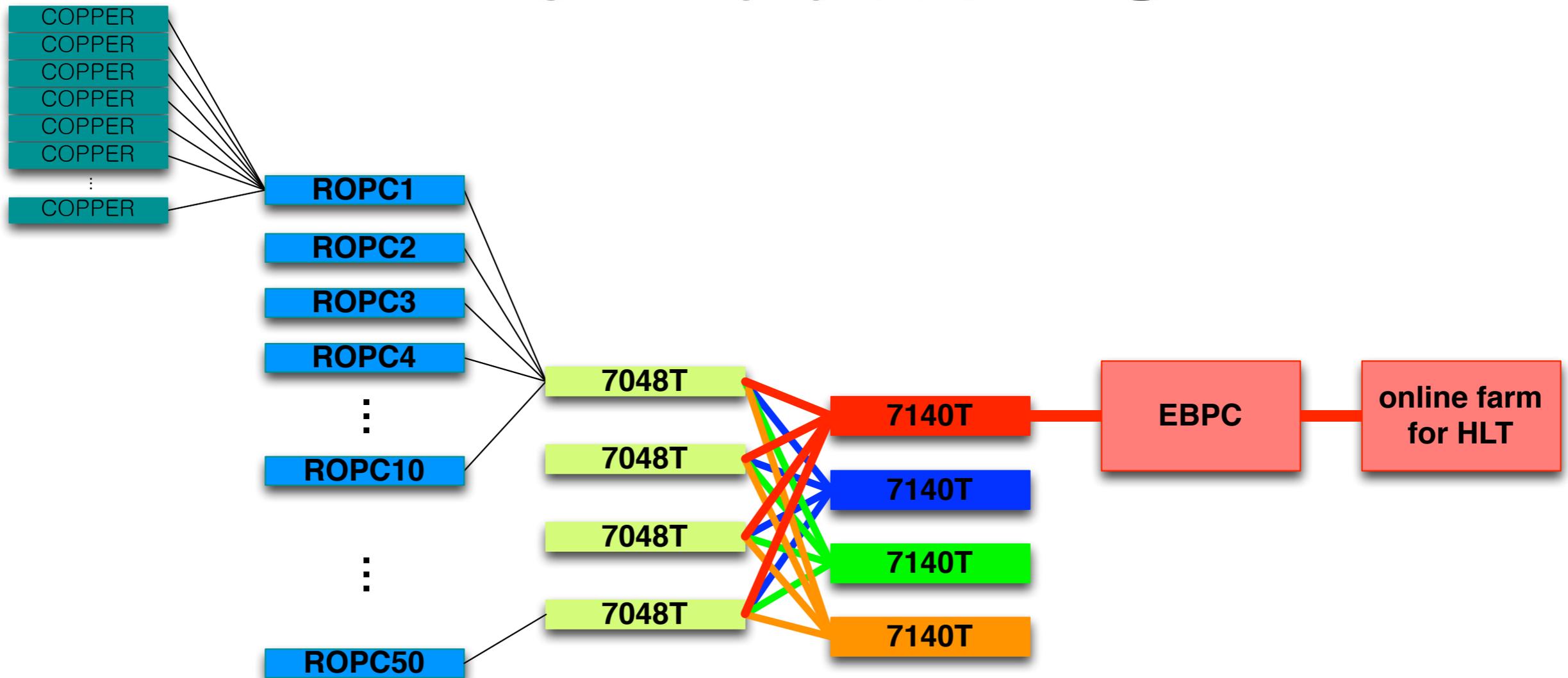
View at 7140T or 7148SX



**7140T handles 10GBase-T and
EBPC can be 50m away from the switch.
Also GbE can be use for debug purpose in emergency.**

**7148SX faster than that 7140T,
but switch and all PC must be closer than 7m.**

View at EBPC



EBPC has
one downstream for online farm
and 50 upstreams for each ROPC.

EBPC handles,

- Single downstream
 - Accept single connection from top of online farm
 - process spawned by xinetd in the accept timing
- Multiple upstream
 - connects to all ROPCs when it started
 - Addresses of ROPCs are obtained from configuration file

Before starting RUN (ONLINE READY)

ROPC1

ROPC2

ROPC3

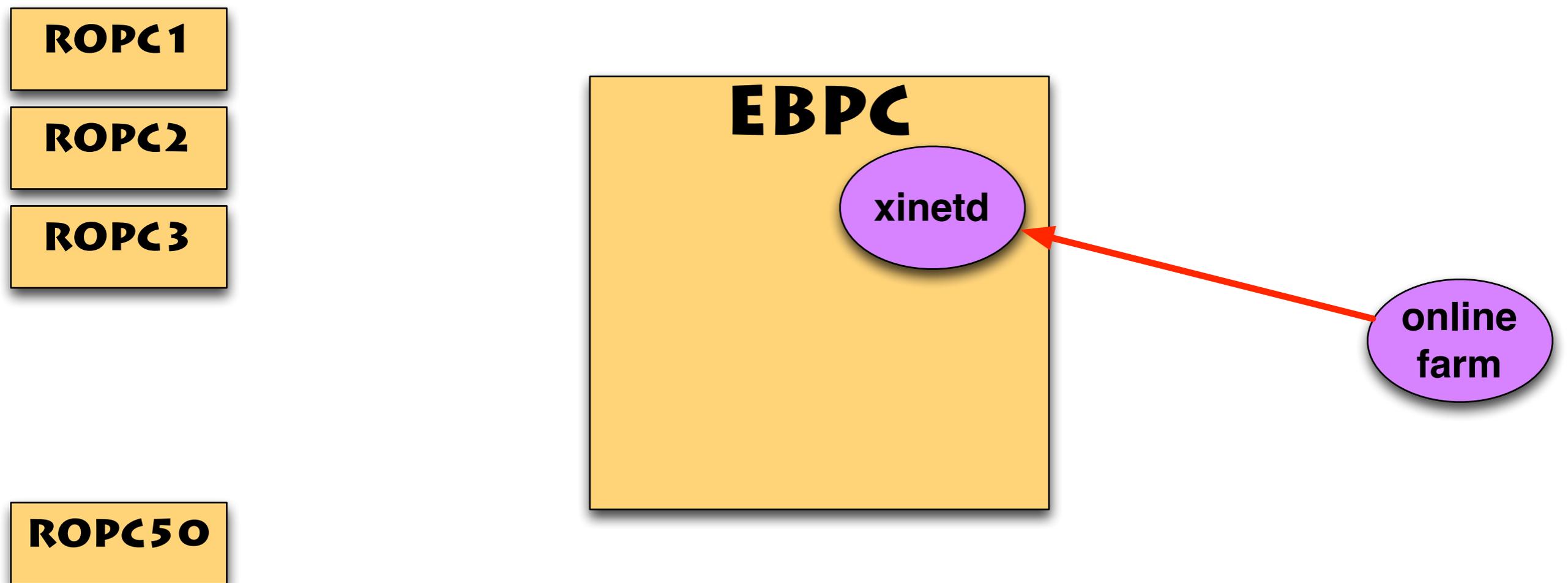
ROPC50

EBPC

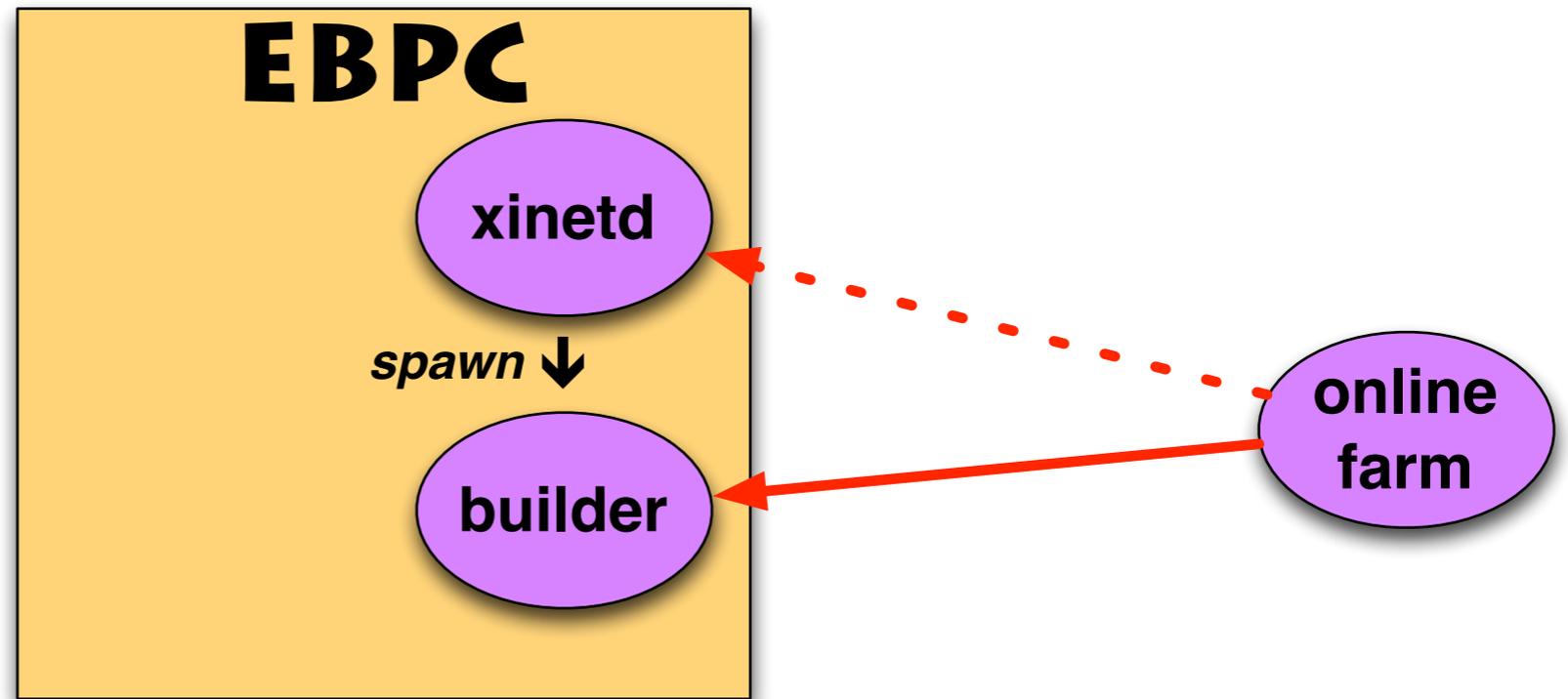
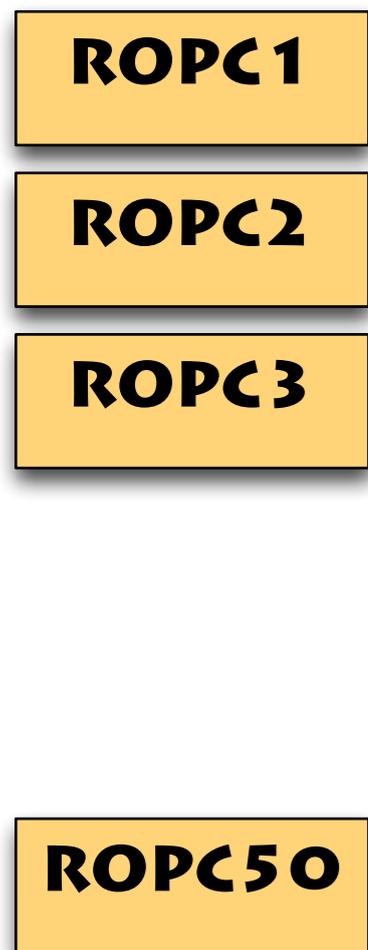
xinetd

online
farm

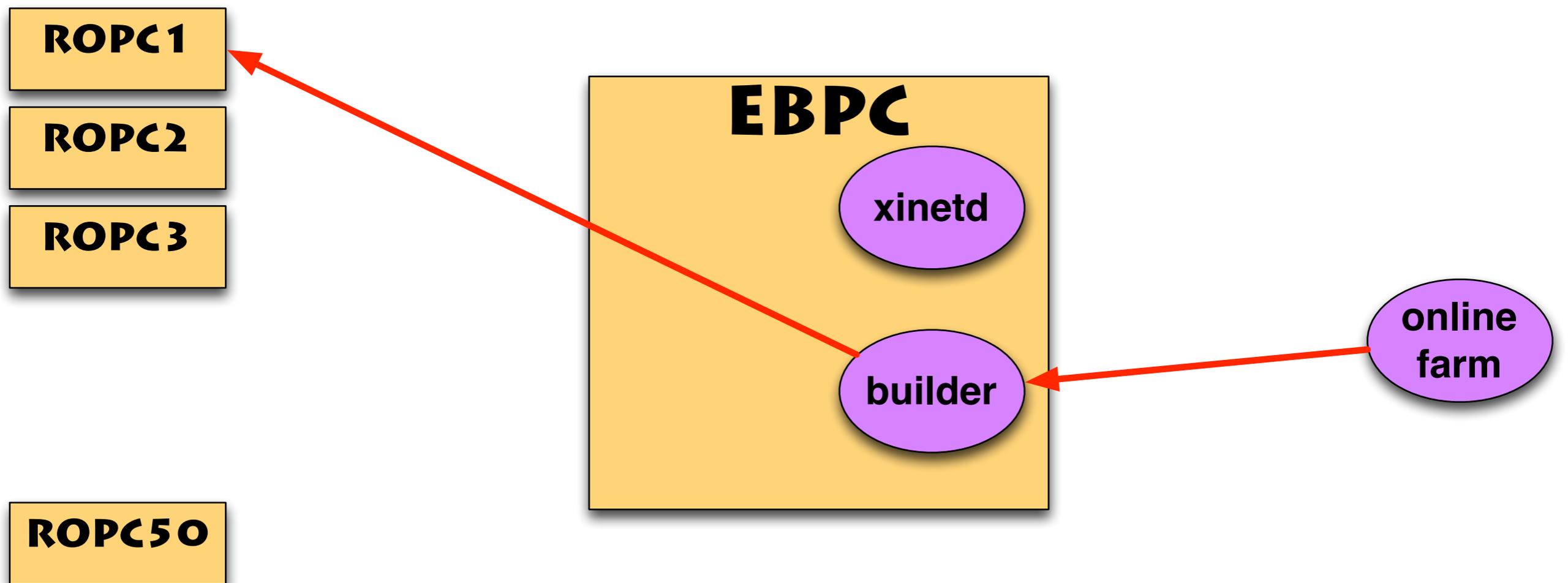
Online farm initiates connection to EBPC



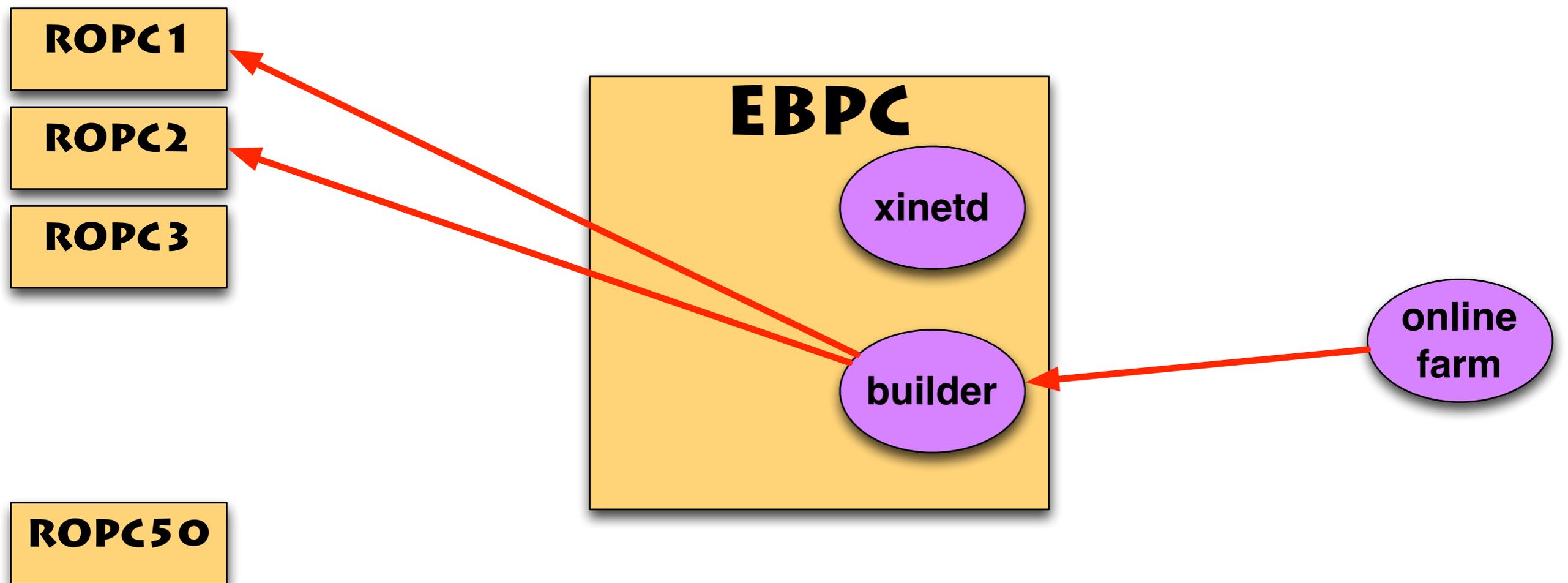
xinetd spawns actual EB program (ONLINE TRANS)



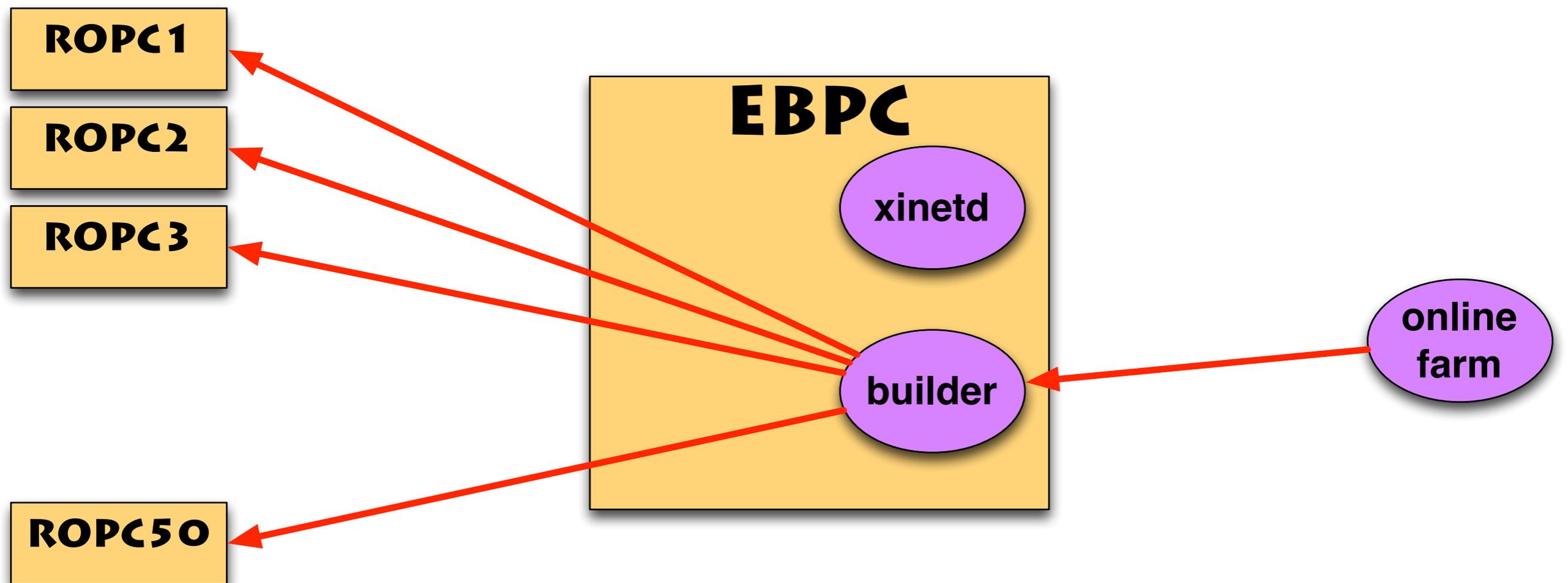
Actual program connect to all of COPPERS



(ONLINE TRANS)



Still ONLINE TRANS
until all ROPCs have successfully
connected to all COPPERS



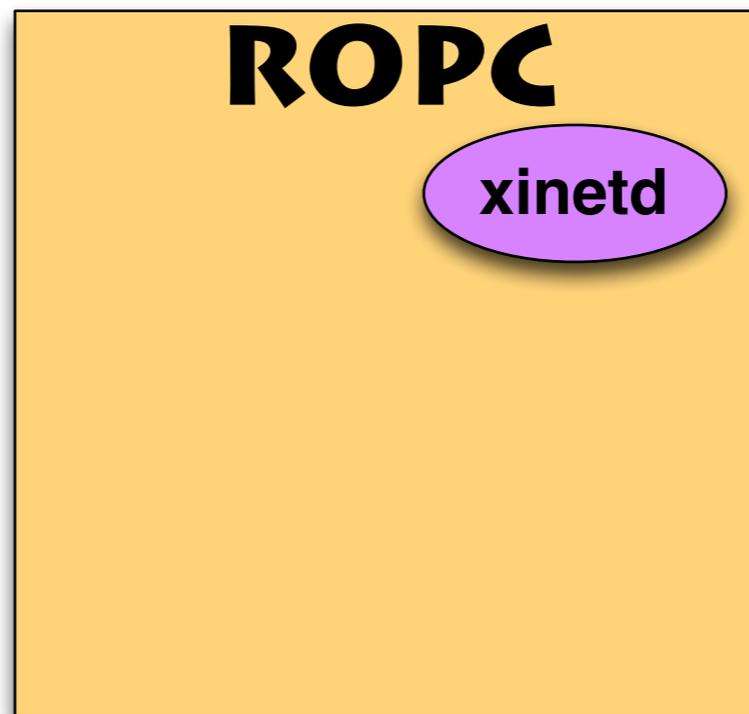
ROPC handles,

- Multiple downstream
 - All EBPC will connect me.
 - # of downstream connection is obtained by configuration file
- Multiple upstream
 - All COPPER must be connected by me
 - # of COPPER is obtained by configuration file
 - orrwho?
- Connects to COPPERs **after** all downstream become ready
- also spawned by xinetd on ROPC

Before start RUN (ONLINE READY)

- COPPER 1**
- COPPER 2**
- COPPER 3**
- COPPER 4**
- COPPER 5**
- COPPER 6**

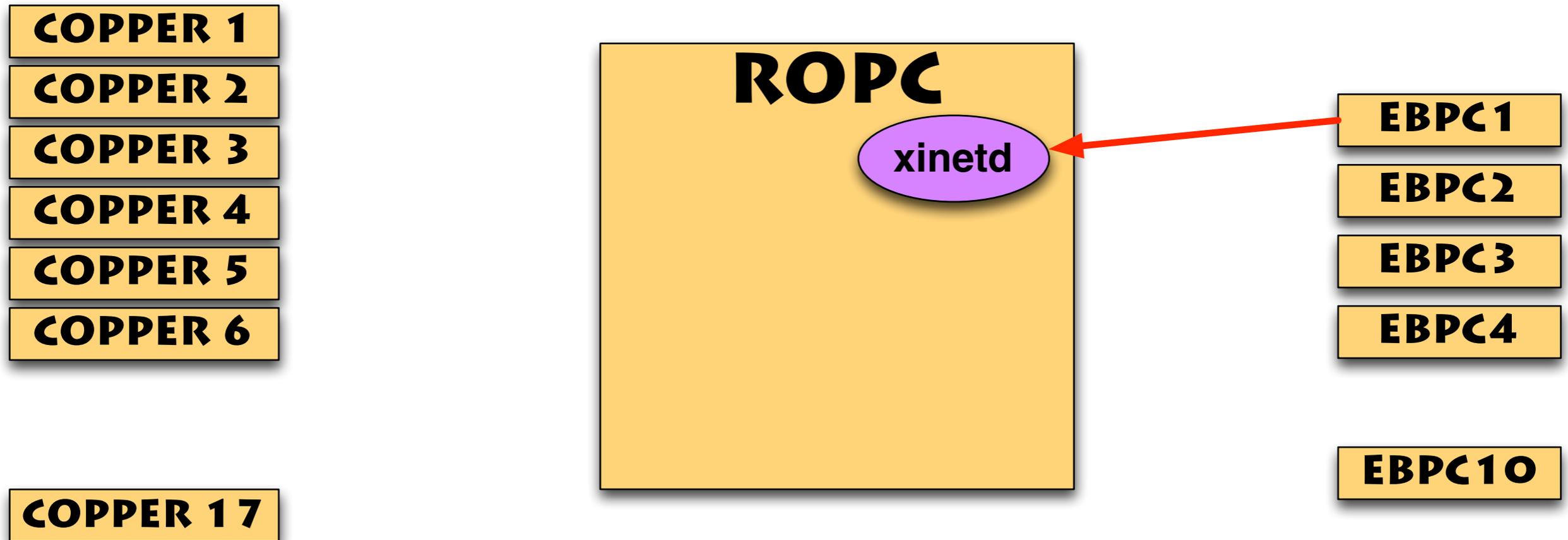
- COPPER 17**



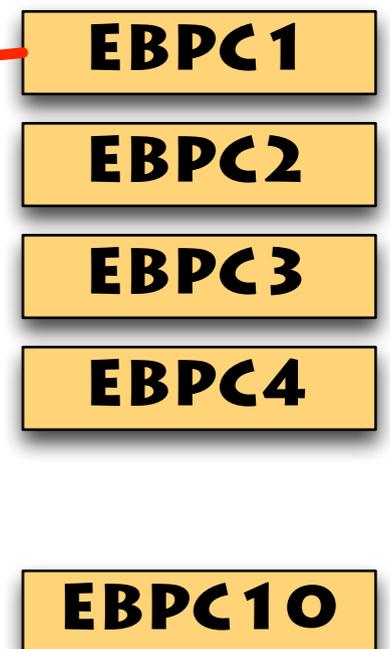
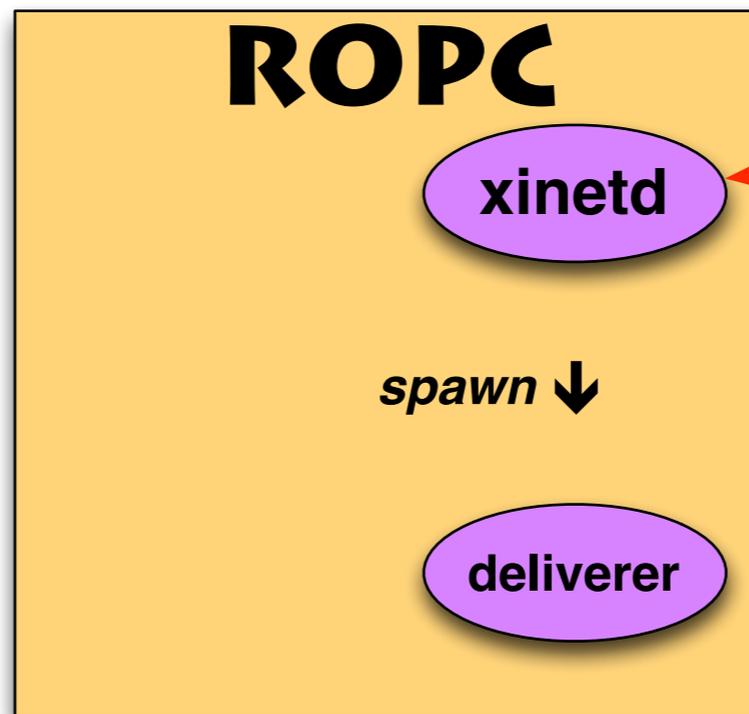
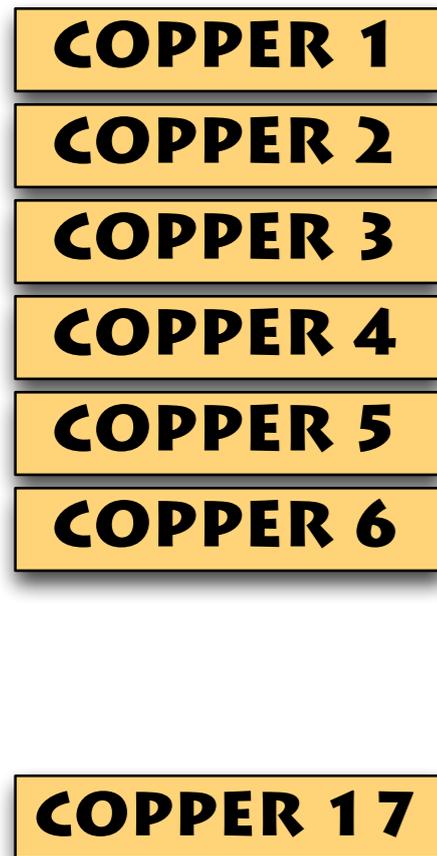
- EBPC1**
- EBPC2**
- EBPC3**
- EBPC4**

- EBPC10**

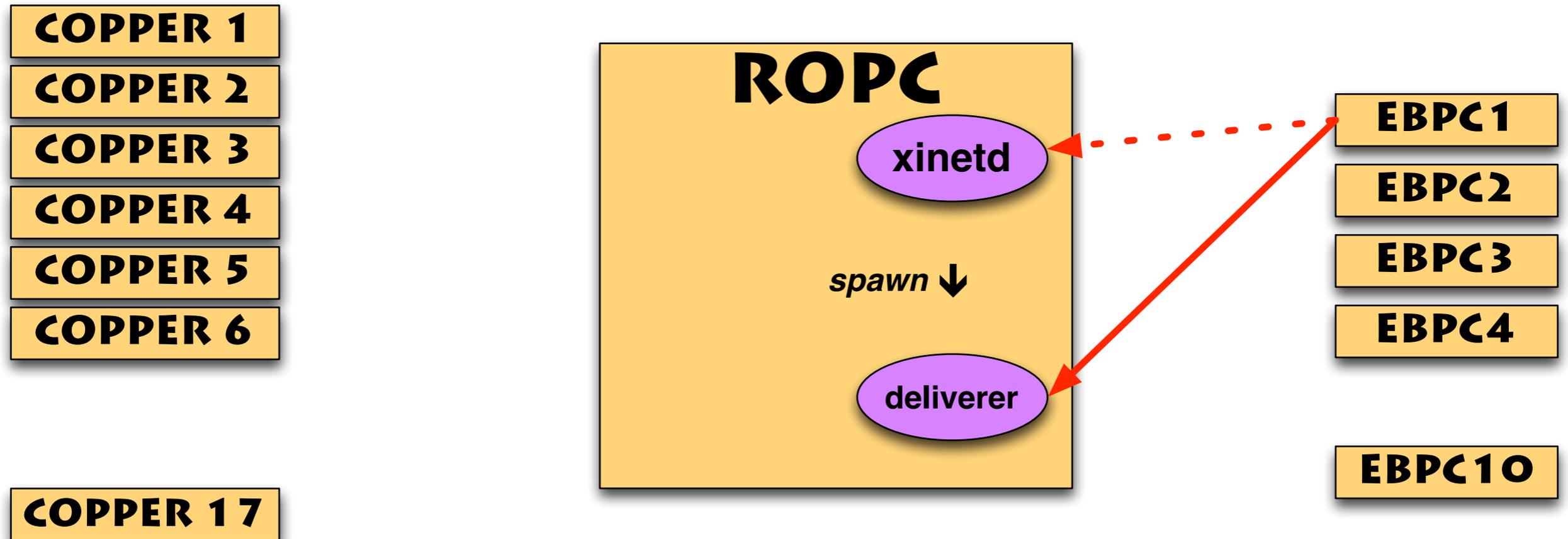
xinetd accepts 1st EBPC.
Already ONLINE TRANS because of
EBPC start procedure



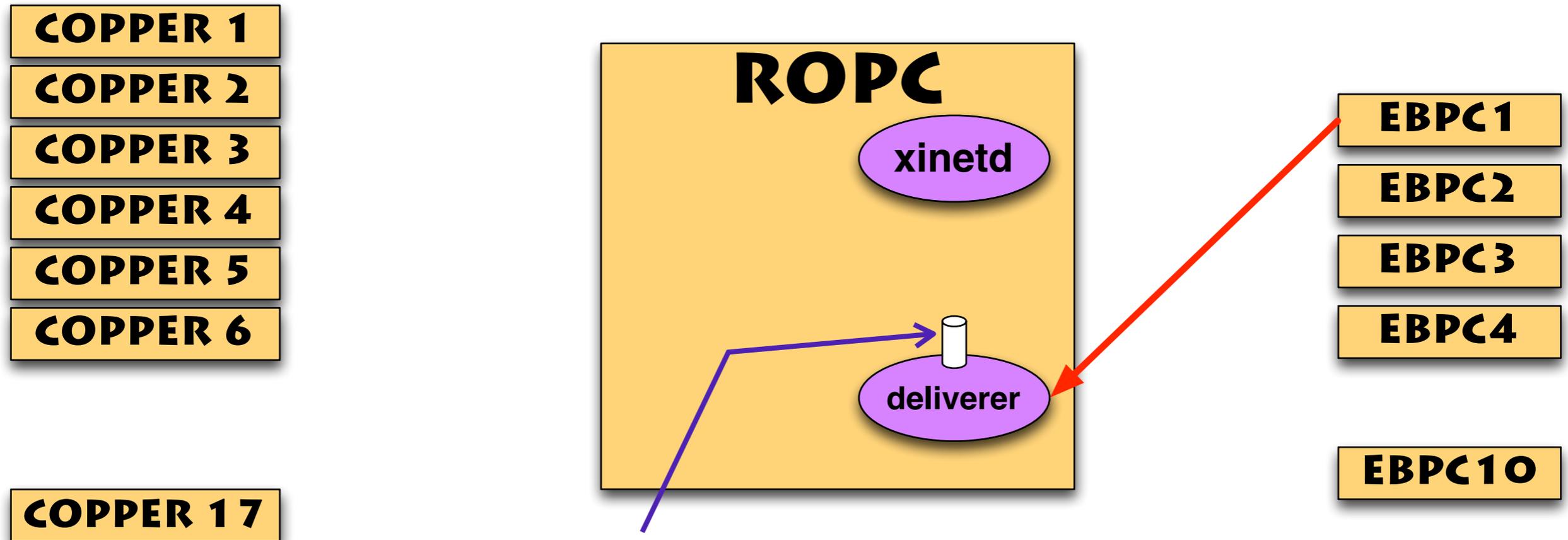
xinetd spawns "deliverer"



connection is handled by deliverer



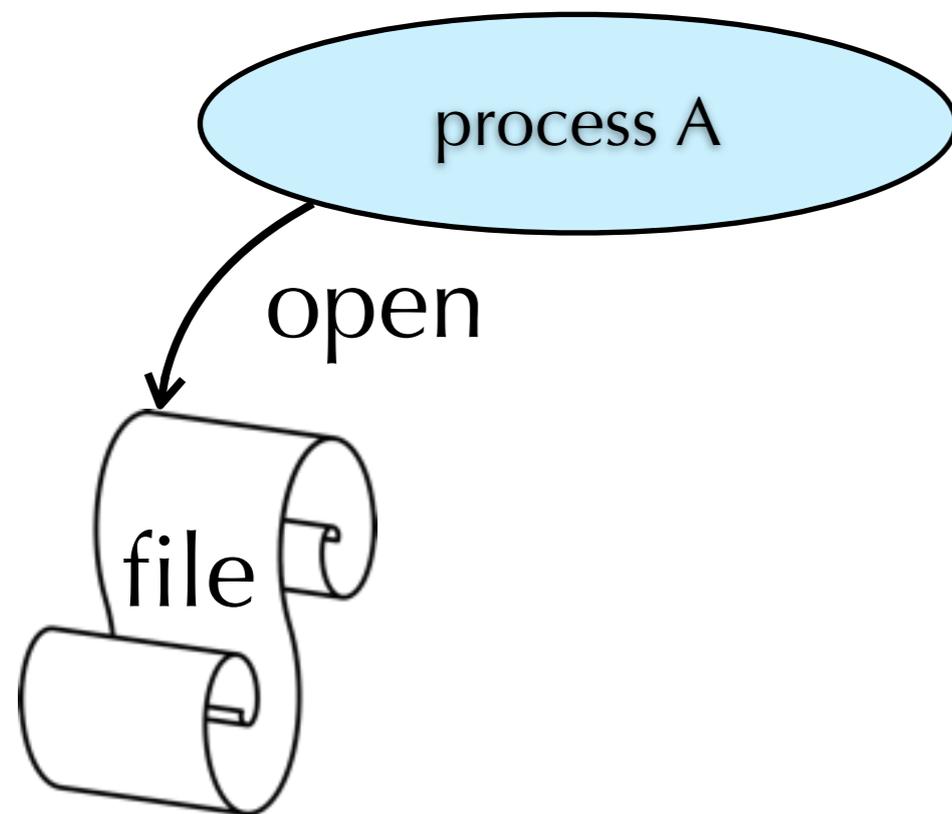
1st deliverer: "Oh, I'm 1st deliverer.
I'll open an UNIX domain socket for
SCM_RIGHTS.



UNIX domain socket
to receive file descriptor.
network socket can be passed via here.

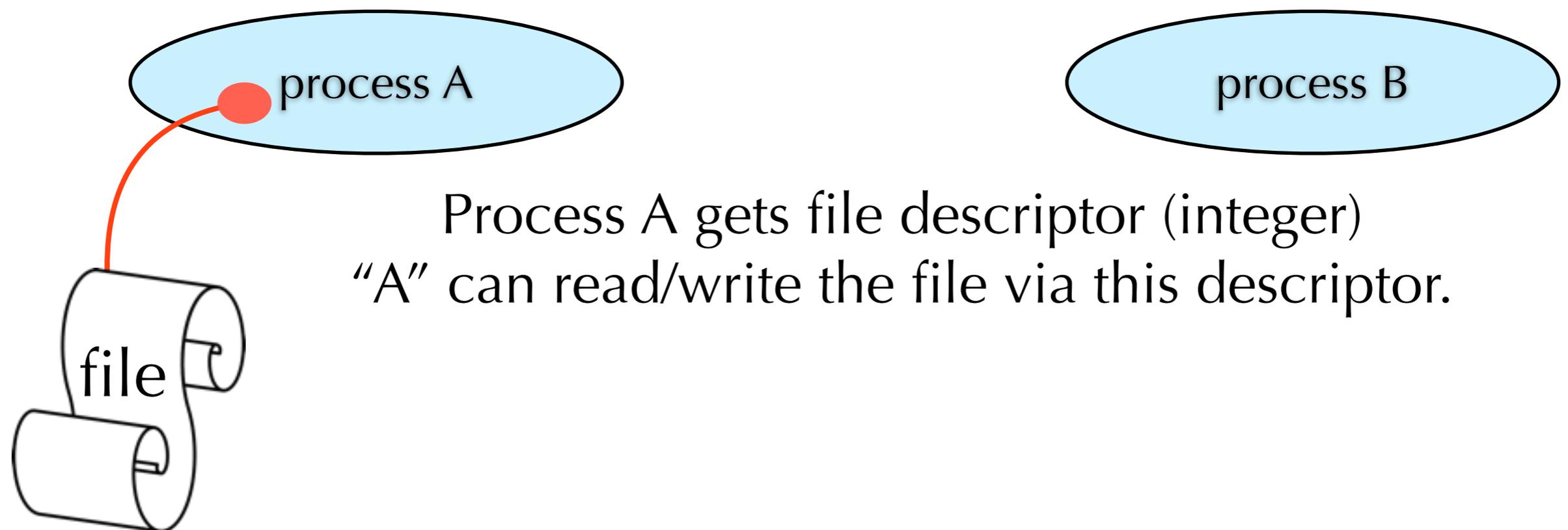
SCM_RIGHTS + sendmsg/rcv

- Mechanism to pass “file descriptor” to other process



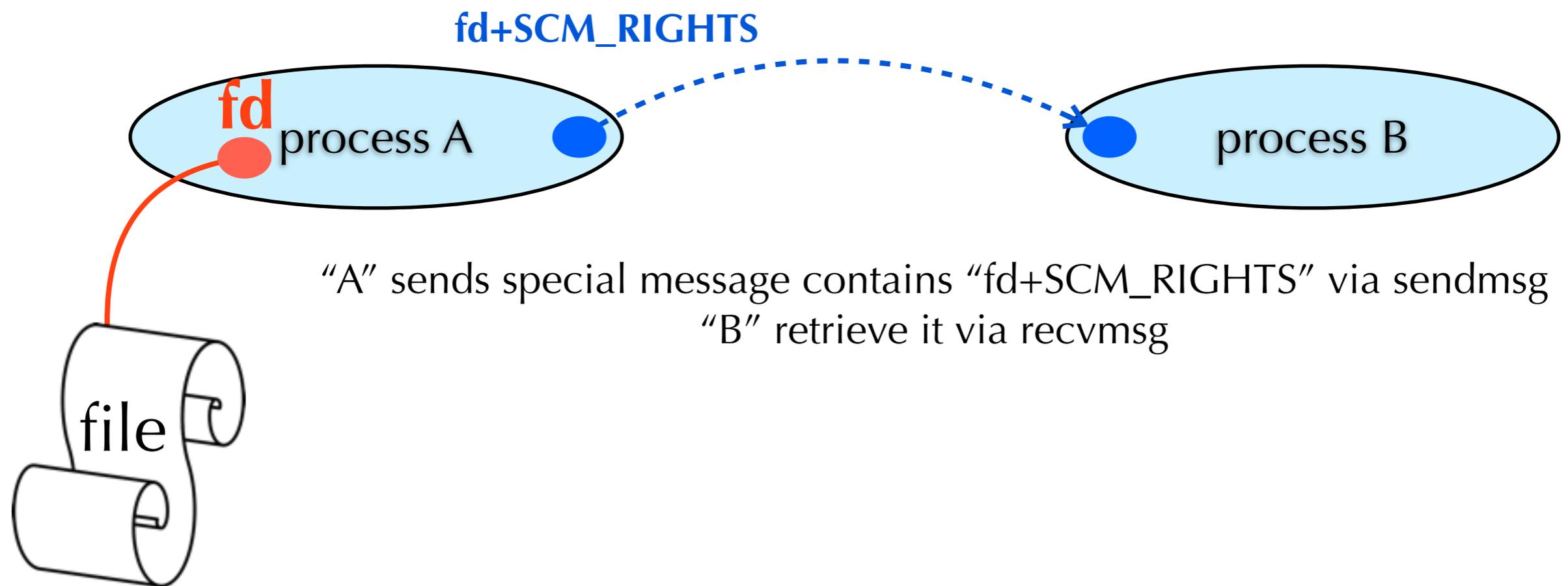
SCM_RIGHTS + sendmsg/rcv

- Mechanism to pass “file descriptor” to other process



SCM_RIGHTS + sendmsg/recvmsg

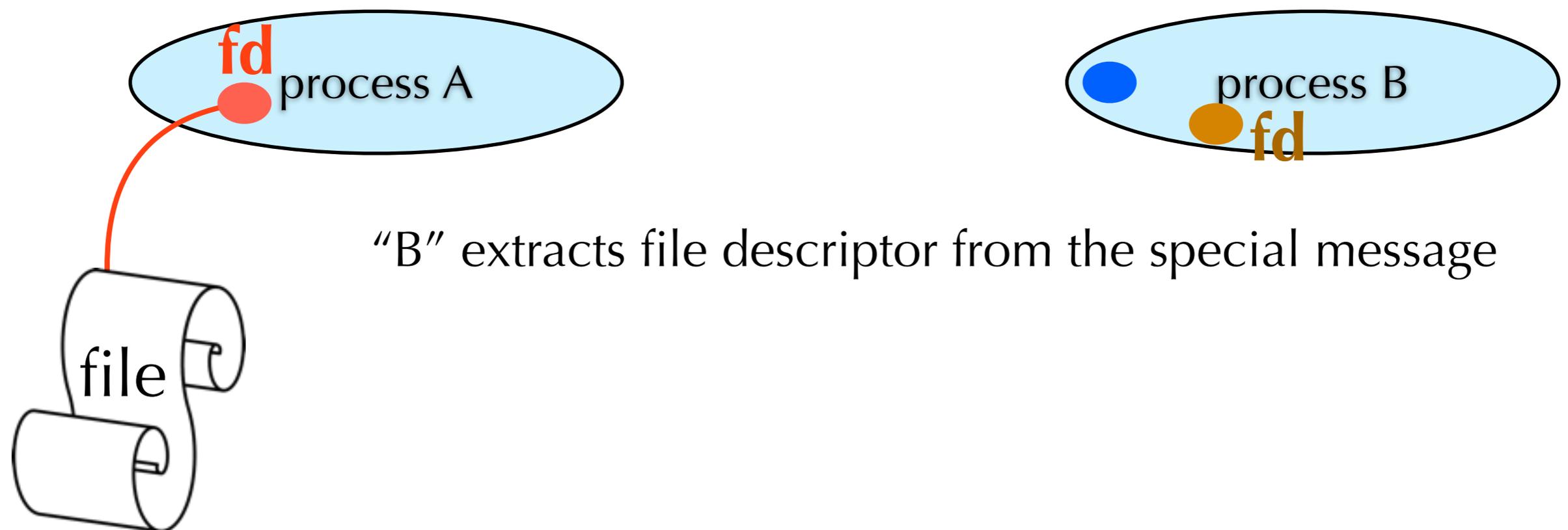
- Mechanism to pass “file descriptor” to other process



SCM_RIGHTS + sendmsg/rcv

- Mechanism to pass “file descriptor” to other process

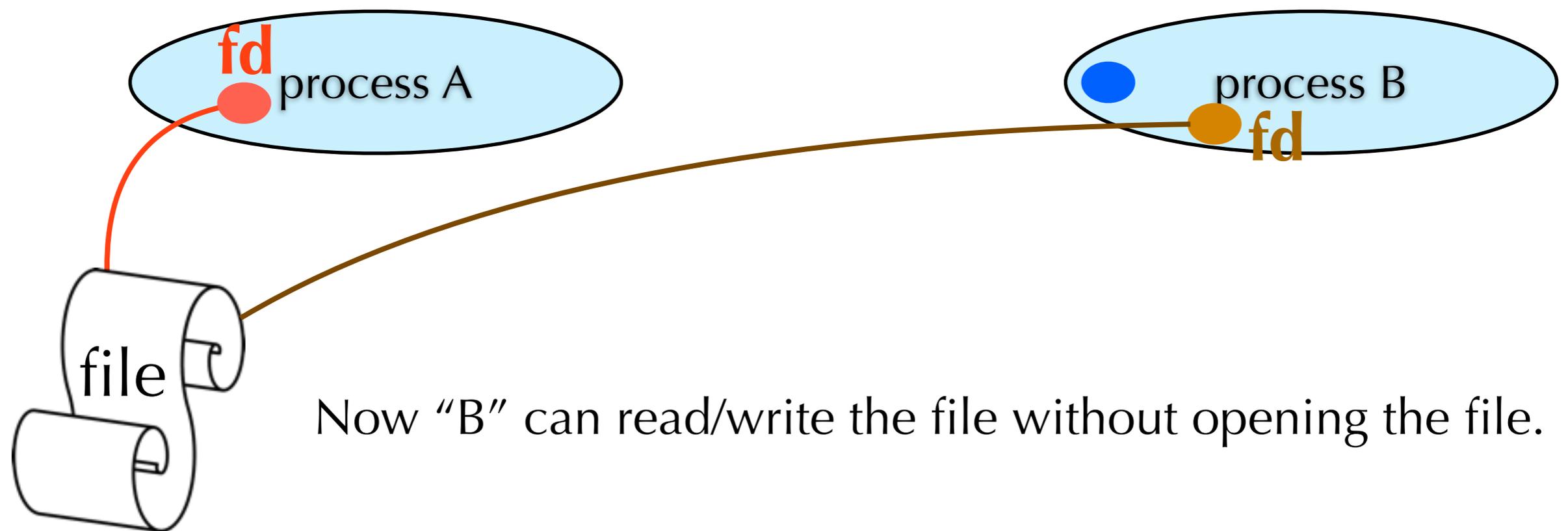
fd+SCM_RIGHTS



“B” extracts file descriptor from the special message

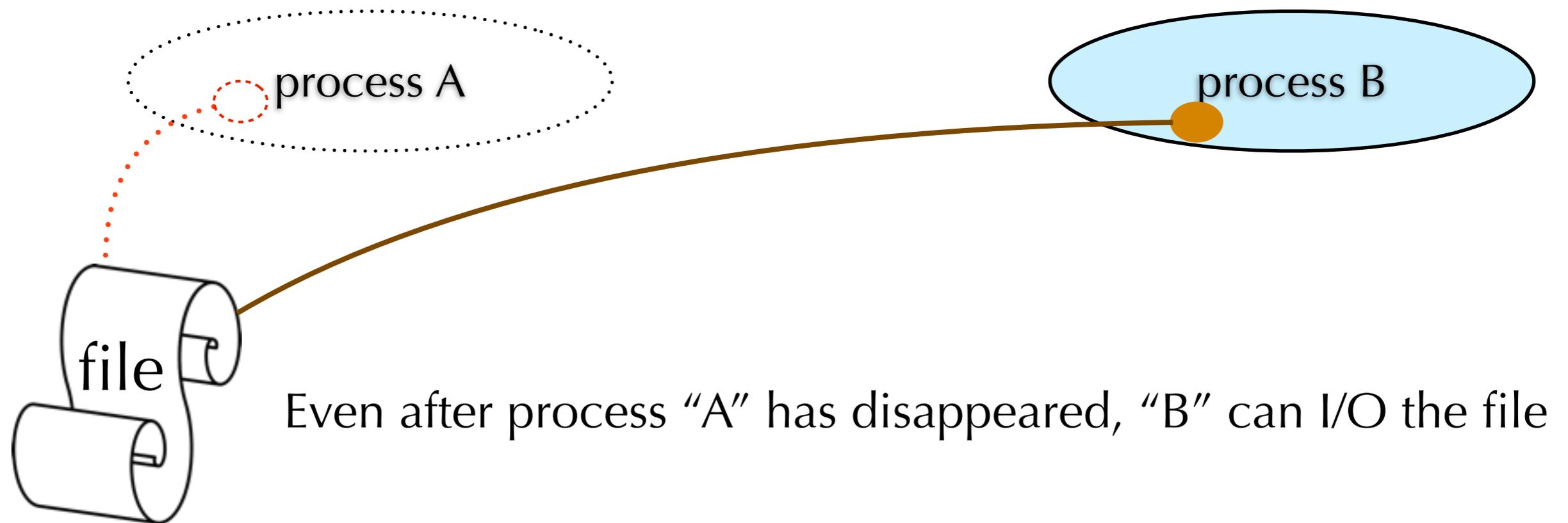
SCM_RIGHTS + sendmsg/rcv

- Mechanism to pass “file descriptor” to other process

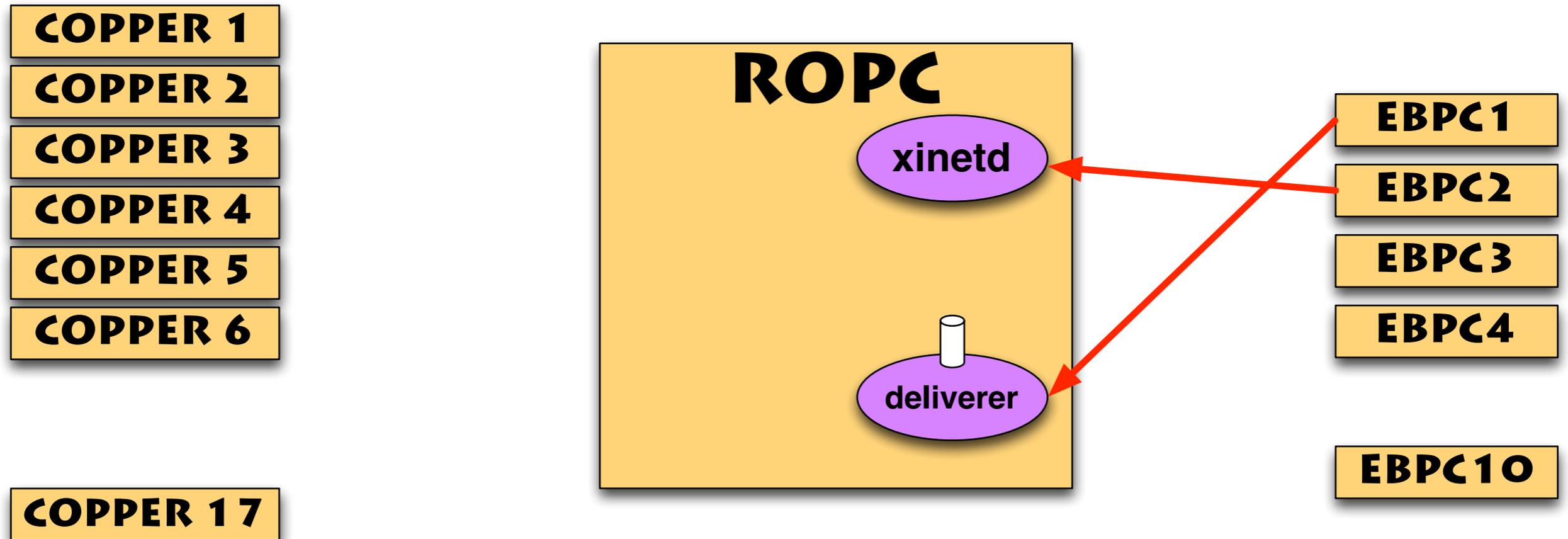


SCM_RIGHTS + sendmsg/rcv

- Mechanism to pass “file descriptor” to other process



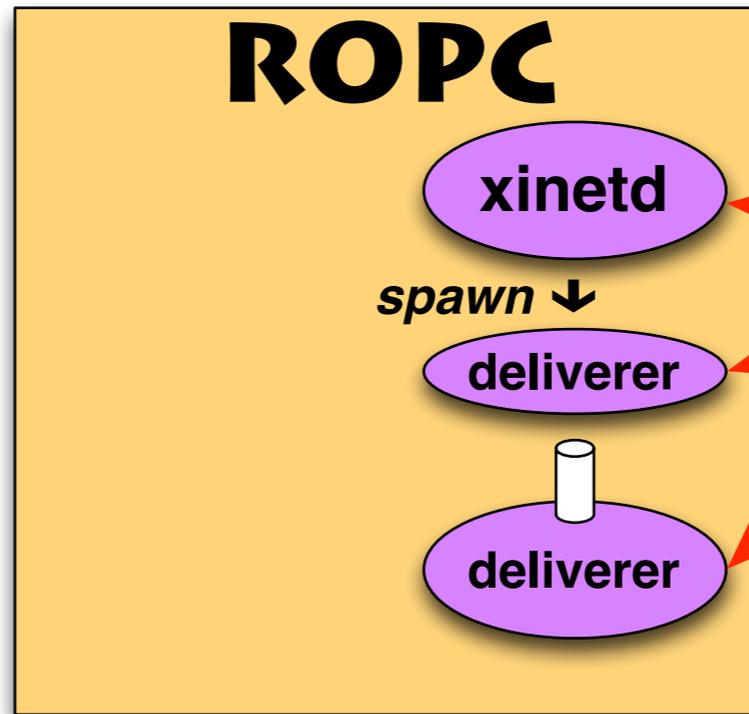
xinetd accepts next connection from EBPC



spawns 2nd deliverer

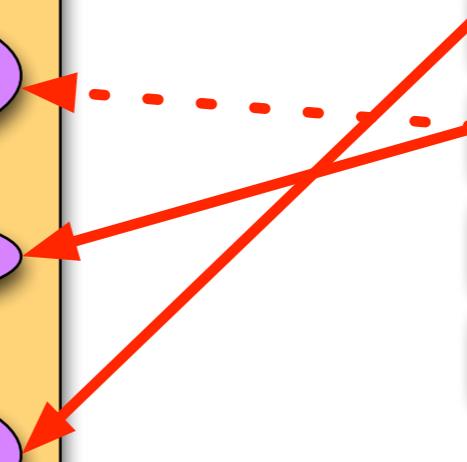
- COPPER 1
- COPPER 2
- COPPER 3
- COPPER 4
- COPPER 5
- COPPER 6

- COPPER 17



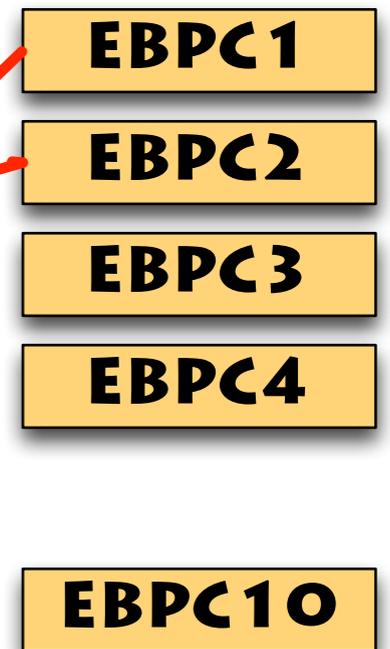
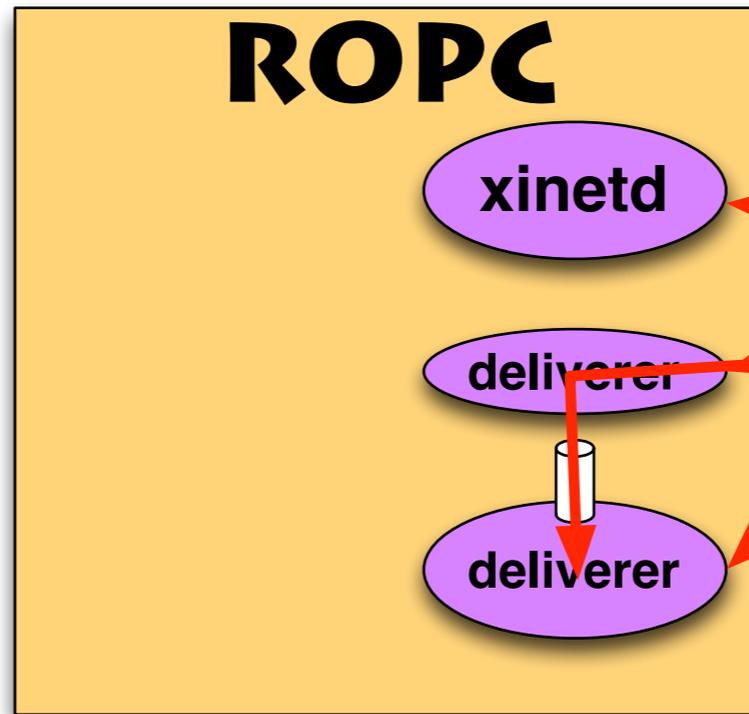
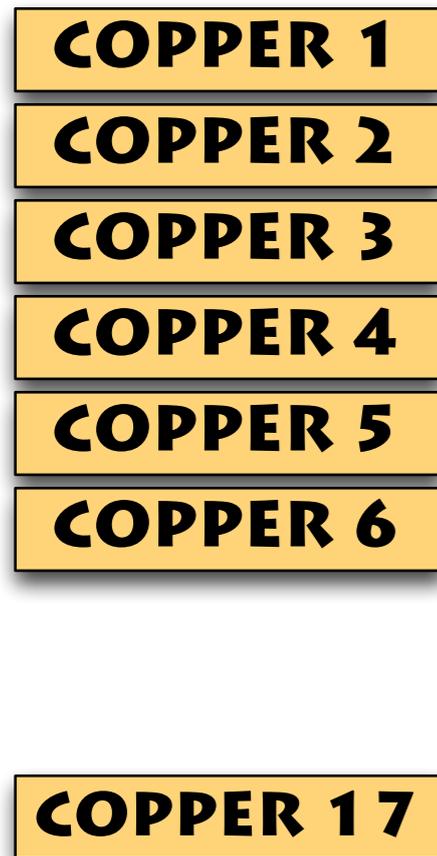
- EBPC1
- EBPC2
- EBPC3
- EBPC4

- EBPC10

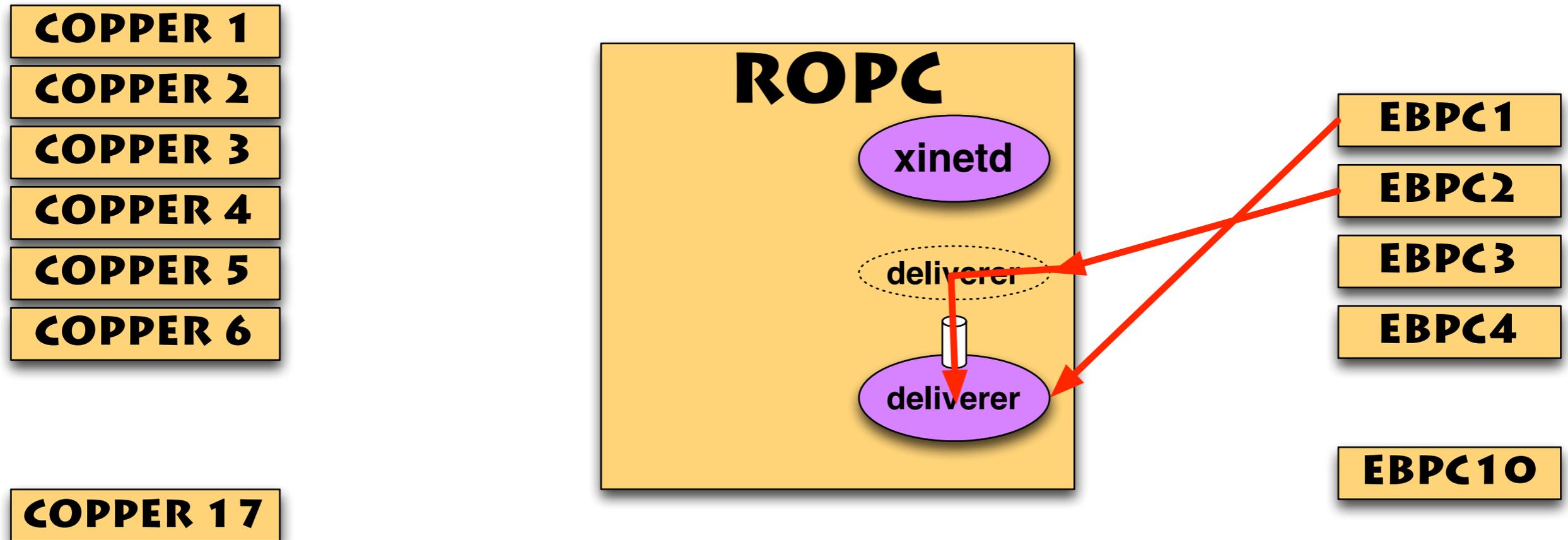


2nd deliverer:

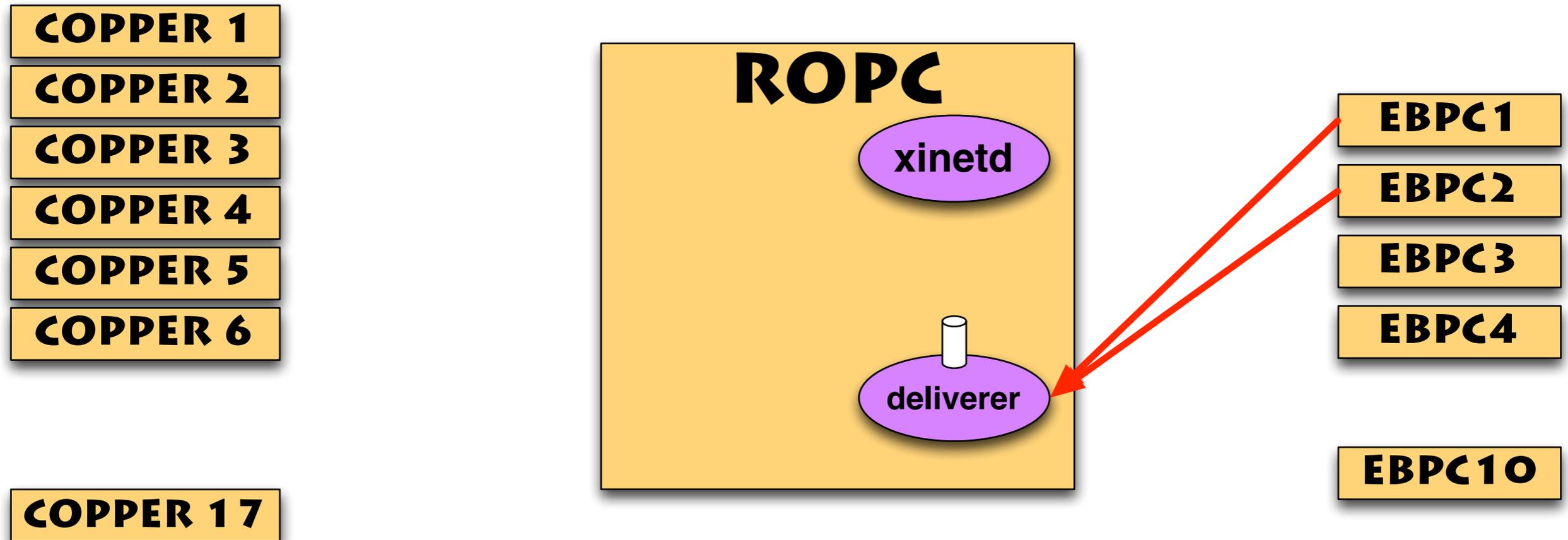
“Oh, already 1st deliver is running.
I can pass my downstream”



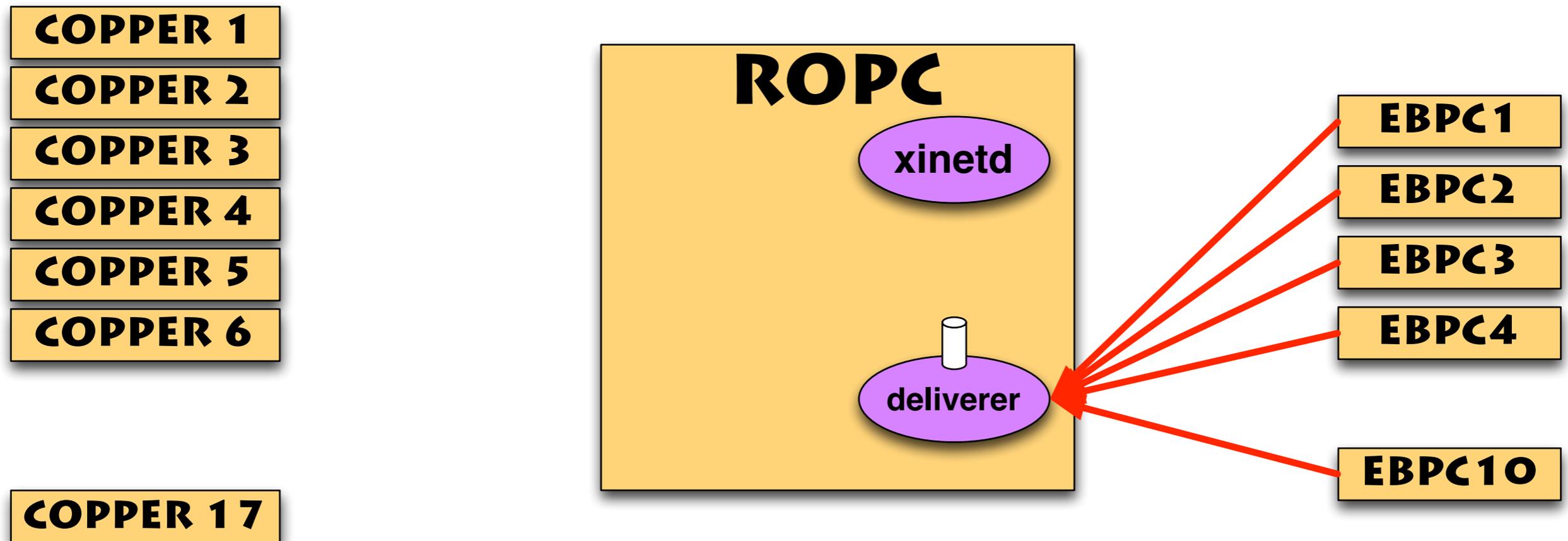
After passing fd, 2nd deliverer exits.



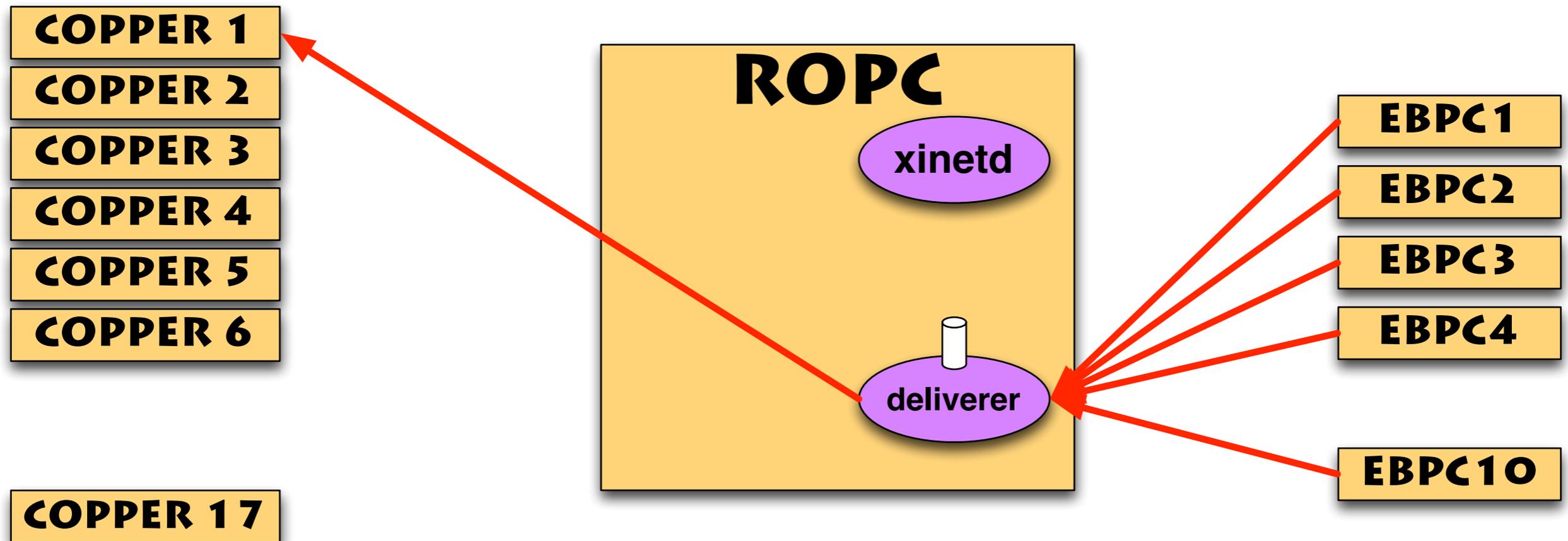
1st deliverer has two connections.



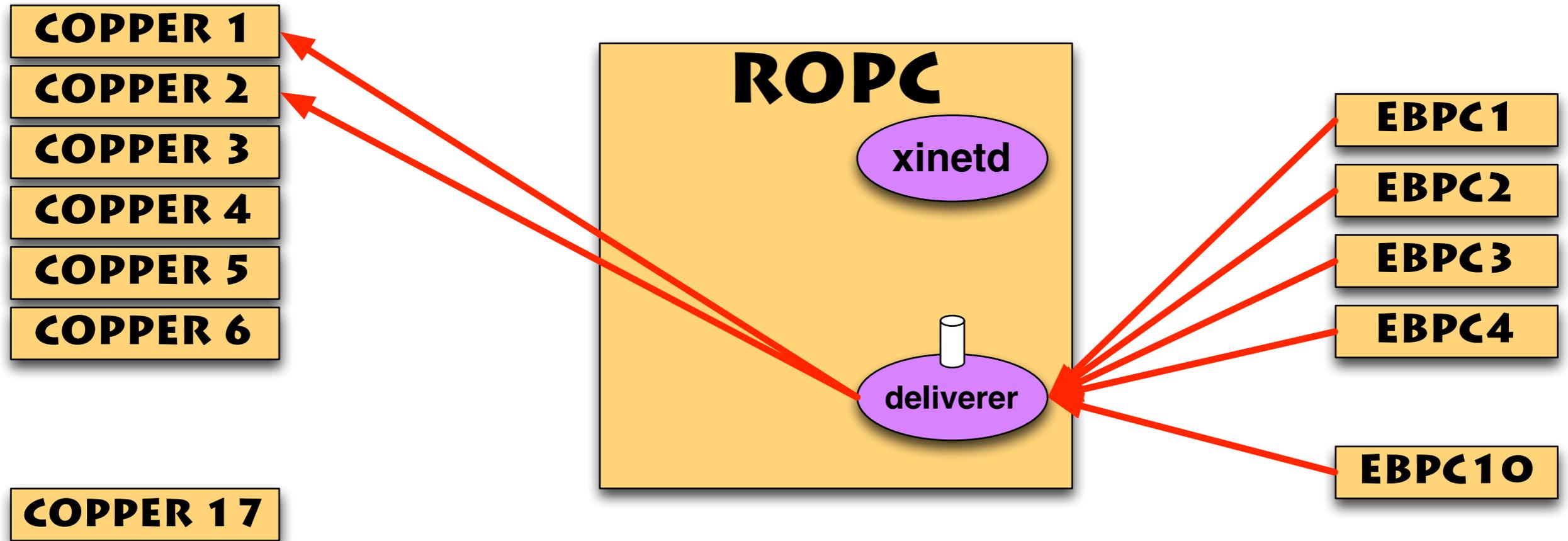
Similarly, all EBPCs are handled by 1st deliverer



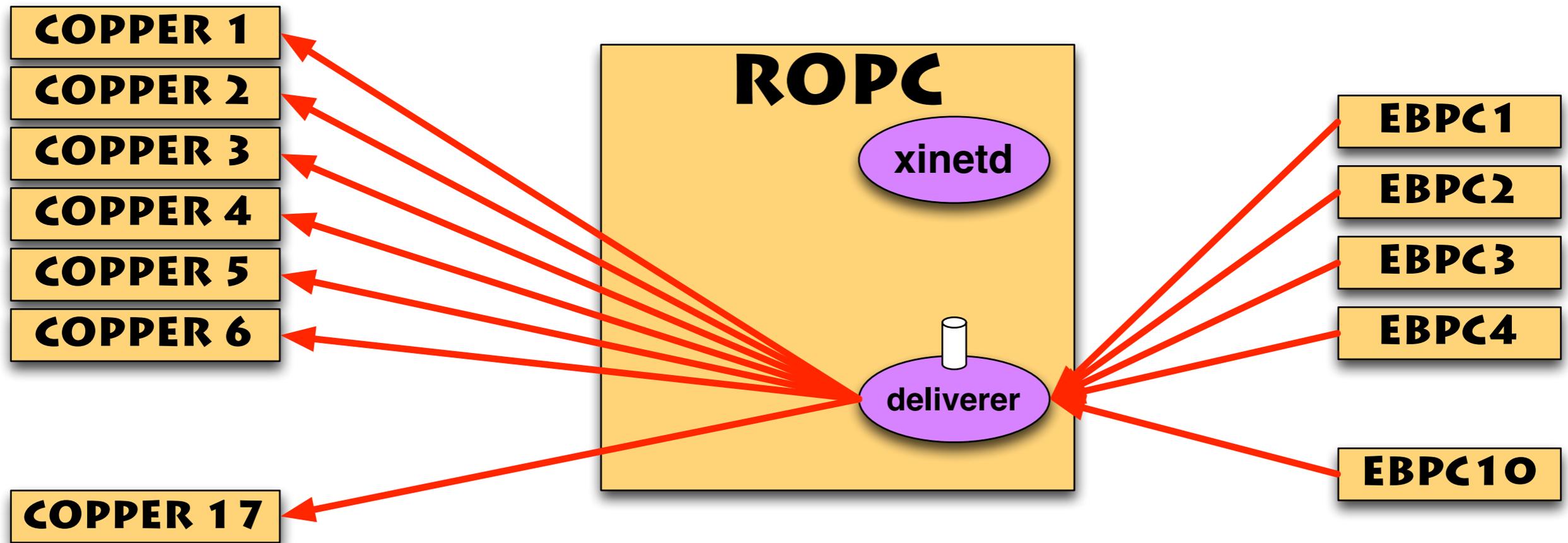
When 1st deliverer gets all EBPCs, begin to connect COPPERS



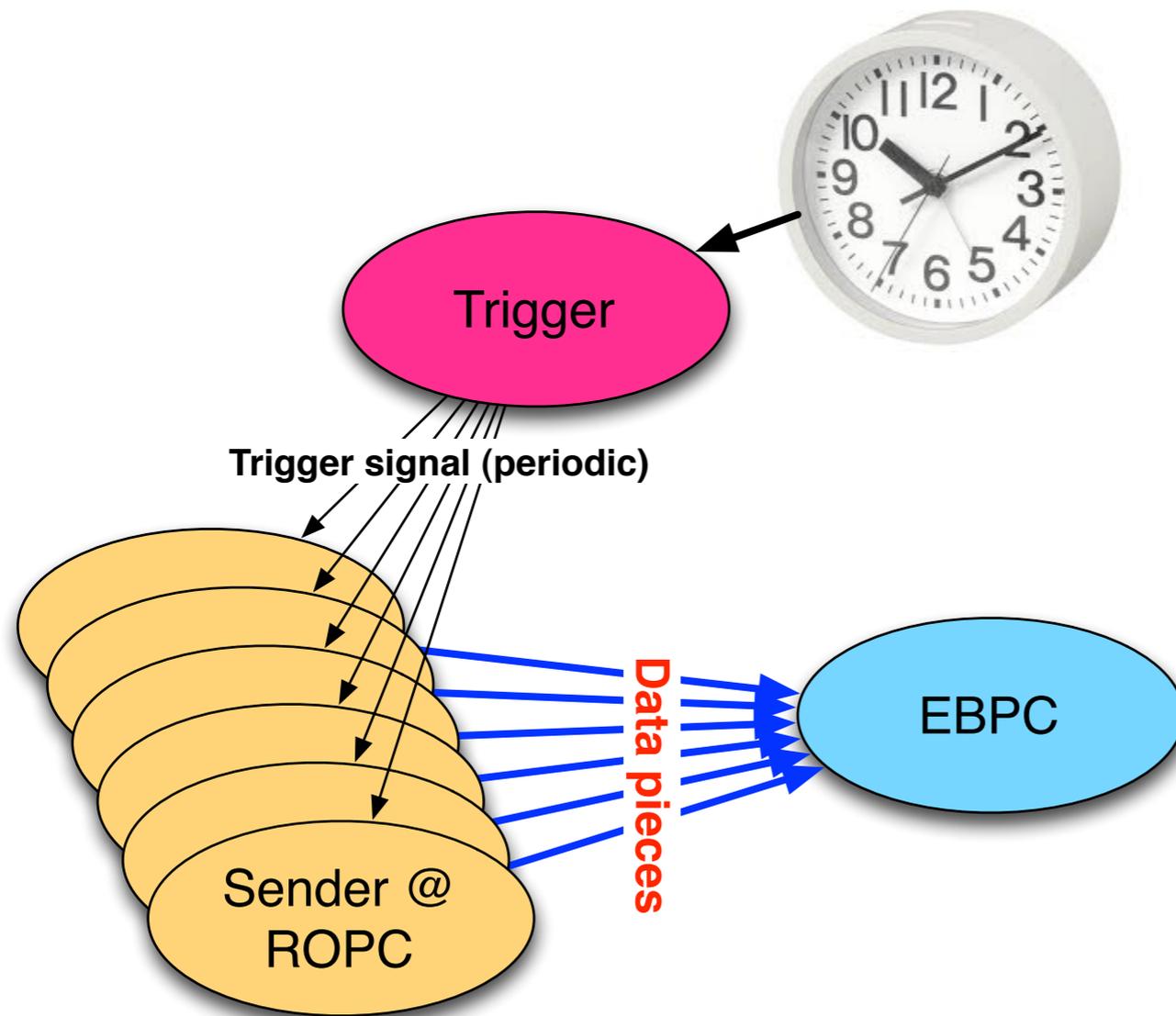
connect to 2nd COPPER,



Now 1st deliverers has all connection for COPPERs and EBPCs.

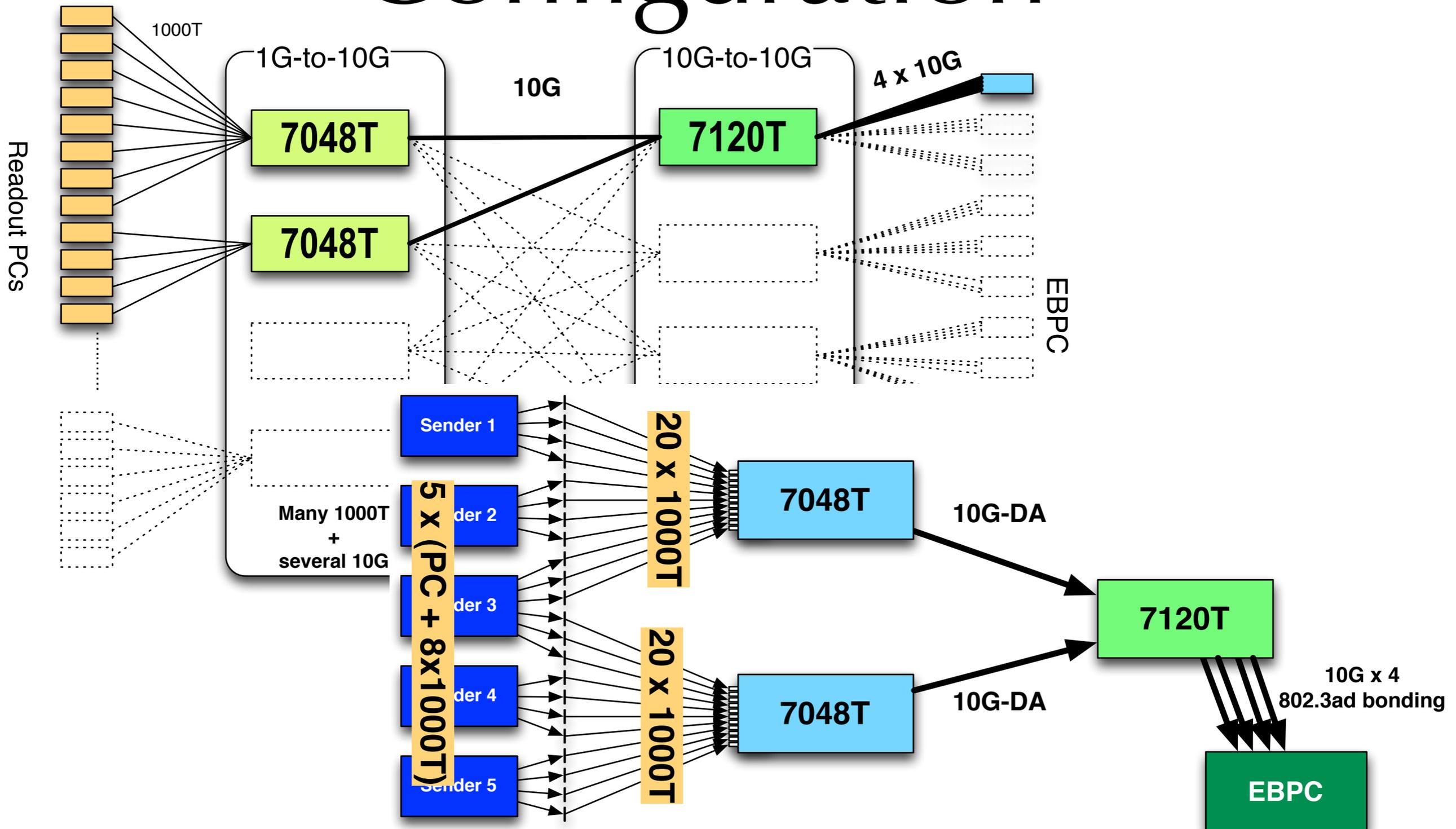


Performance test

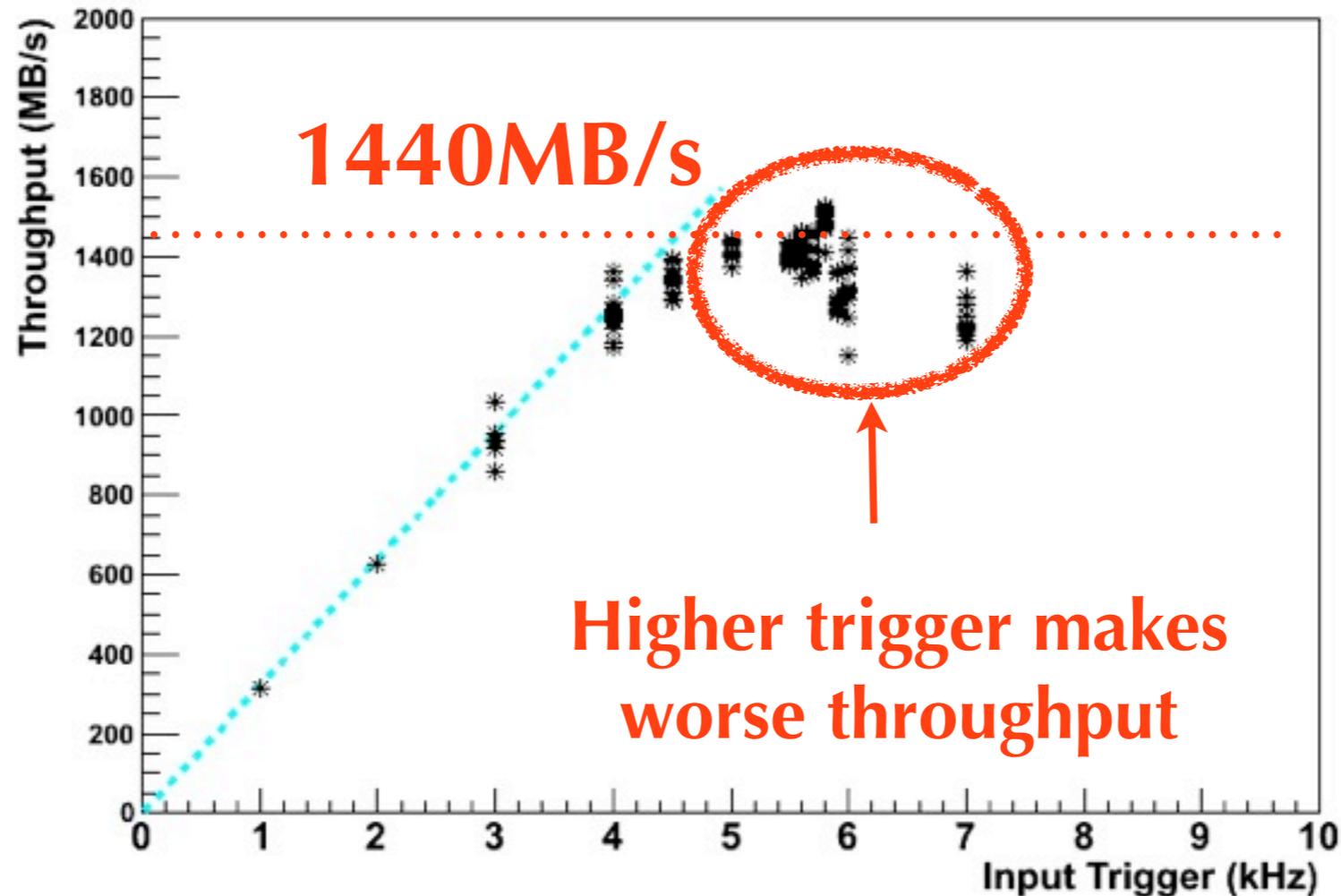


- Trigger sends periodical signal via broadcast
- Sender sends 8kB event fragments via TCP in each trigger
- EBPC concatenates all fragments and discard

Configuration

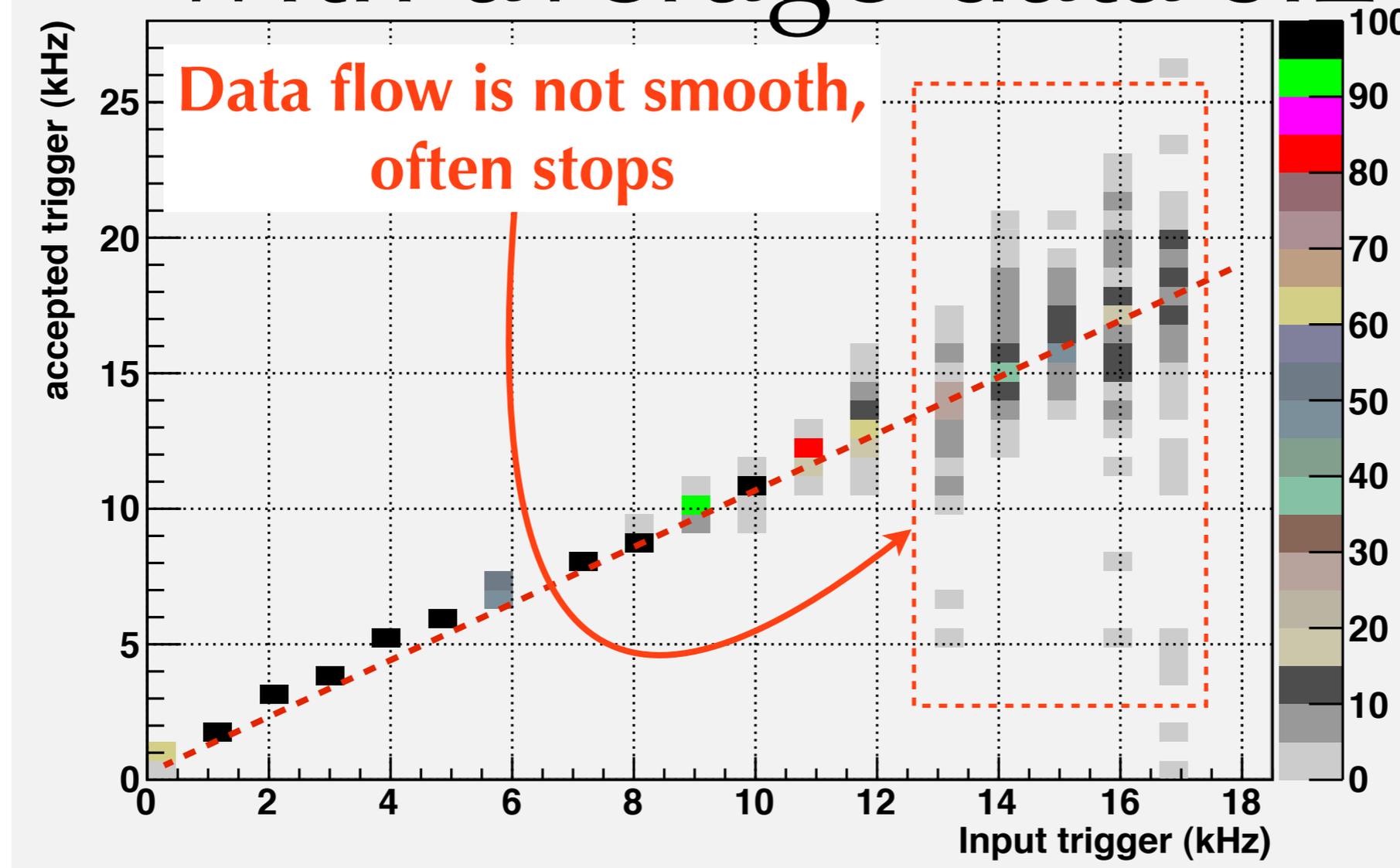


Throughput test with large event size and slow trigger



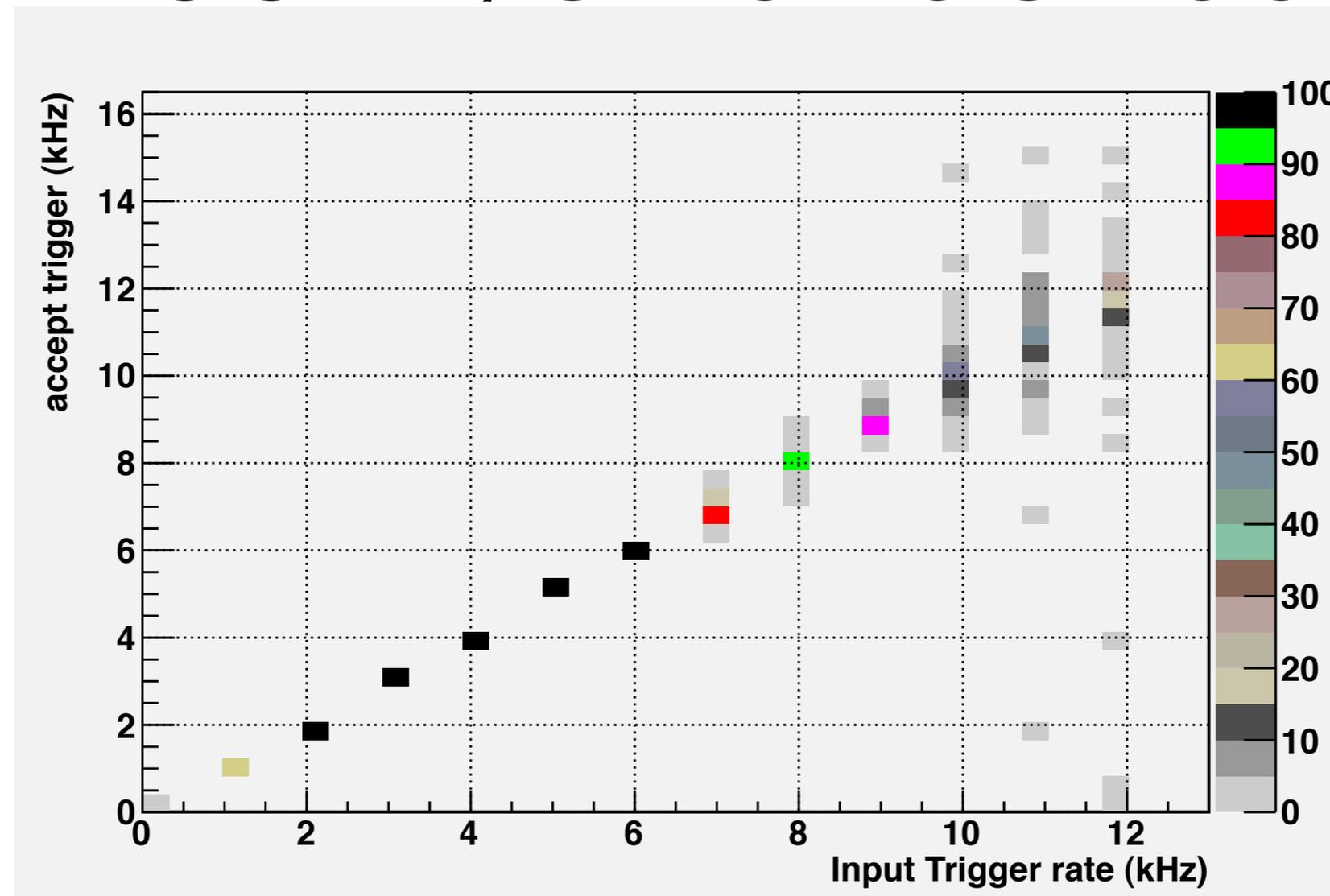
- 2 threads for receiver + 1 thread for event building
 - 4 threads doesn't make faster
- Maximum throughput is about 1440MB/s @ 3.2MB/ev * **4.5kHz**
- **using 10Gx4, 2240MB/s was achieved under 6.8kHz**

Maximum trigger rate with average data size



- fully event data size is 50kB (1280byte per sender)
- 6 threads for receiver + 1 thread for event building
- Maximum throughput is about $50\text{kB} * 14\text{kHz} \Rightarrow 800\text{MB/s}$
- **over 12 - 13kHz is danger zone**

In the case of 120kB/ev (c.f. 100kB/ev under burst)



Over 10kHz is **really** danger zone.

Maximum throughput is still about 1400MB/s

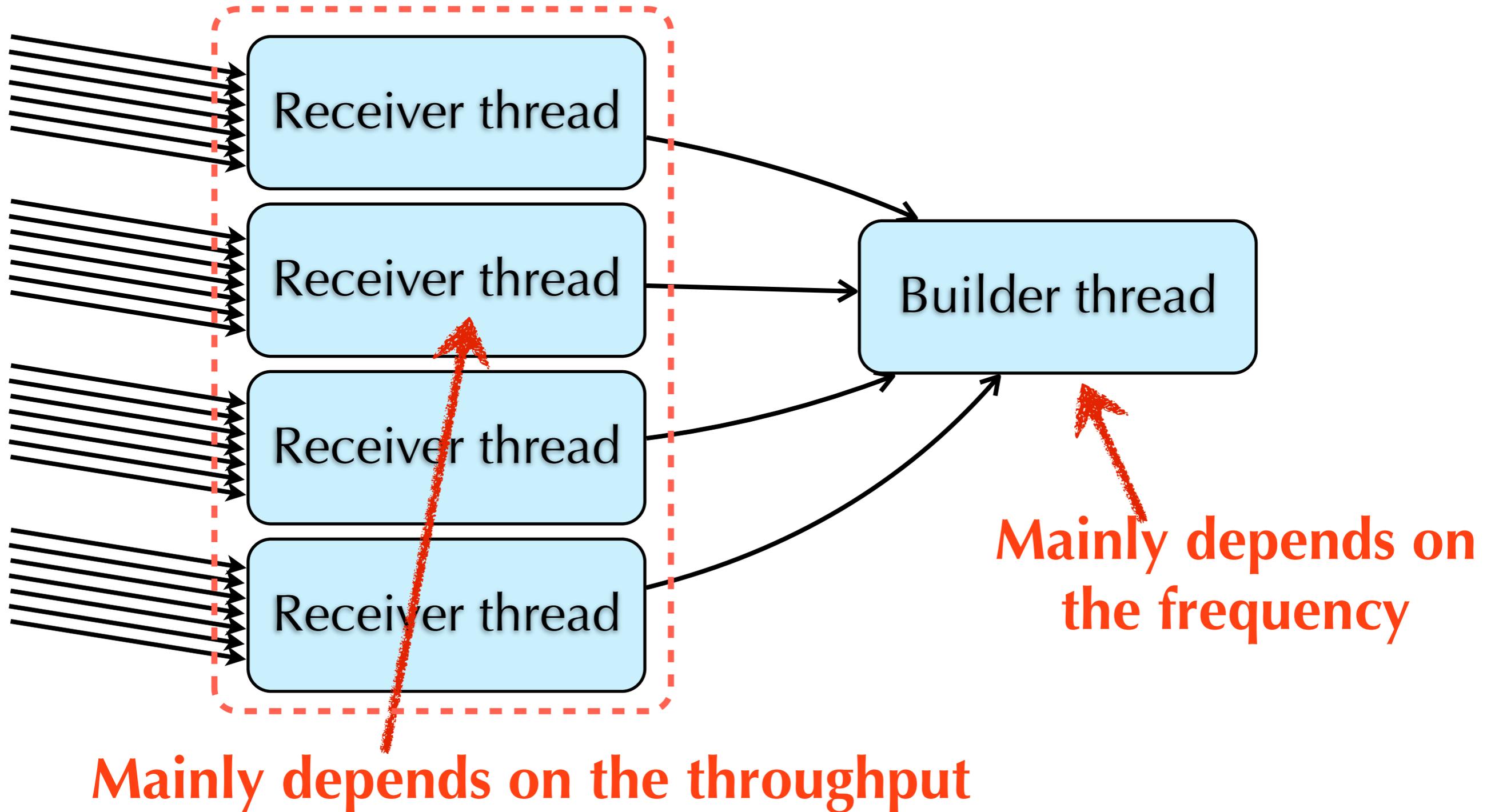
Is this sufficient?

- Maximum accept trigger rate is only 13 - 14kHz per EBPC, because of CPU power.
 - Now surveying where is the bottle neck
- Input rate of E.B.1 is expected as,
 - 50kBytes x 30kHz => 1500MB/s
 - 1500MB/s requires two EBPCs at least, but 30kHz requires three.
 - each EBPC must equips 2~4 links of 10G
- Burst case (dirty beam condition, high background trigger)
 - 100kB x 50kHz => 50GB/s
 - need 5 EBPCs at least

About no-suppression data for calibration

- For the calibration, some detectors want to send data without any reduction to E.B.
- If so, must be clear before purchasing PCs,
 - Event size
 - This affects not only E.B. but also the COPPER driver.
 - How large ?
 - If it is larger than 10kB/ev, maybe problematic
 - Frequency
 - Please do not use all of 50Hz VETO slot!

CPU consumption

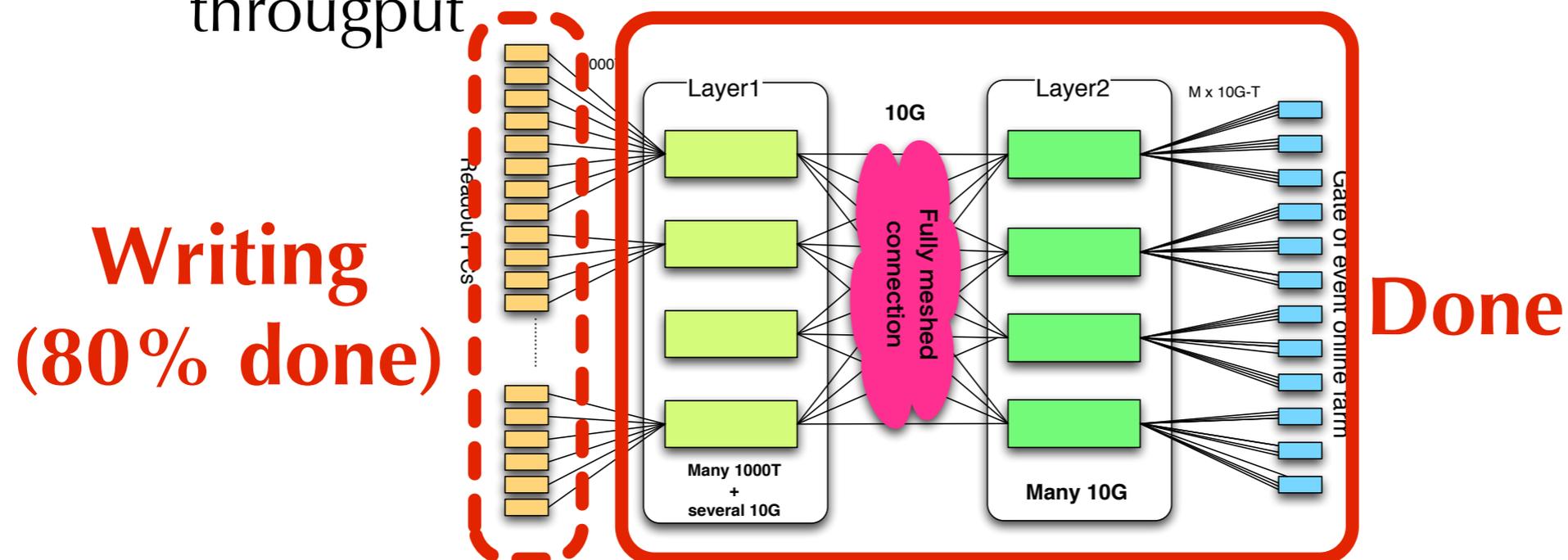


For what, this builder consumes CPU?

- Notification of “I have an event” from receiver to builder
- It uses “pipe”
 - Test of two processes communication “ping-pong” via pipe read/write, achieves 90kHz
 - Now one pipe is assigned to each connection to ROPC
 - => Testbed has 40 pipes, real EBPC will have 50 pipes.
- Other Inter Process Communication methods are needed to be tested for this latency reduction.

Summary of E.B.1

- EBPC must equip multiple links of 10GbE
- For the stable operation, 4 EBPCs and 4 units of online farm is needed.
- Program for EBPC is almost done, for ROPC is still under writing.
- Bottleneck survey in the program for EBPC for the higher throughput



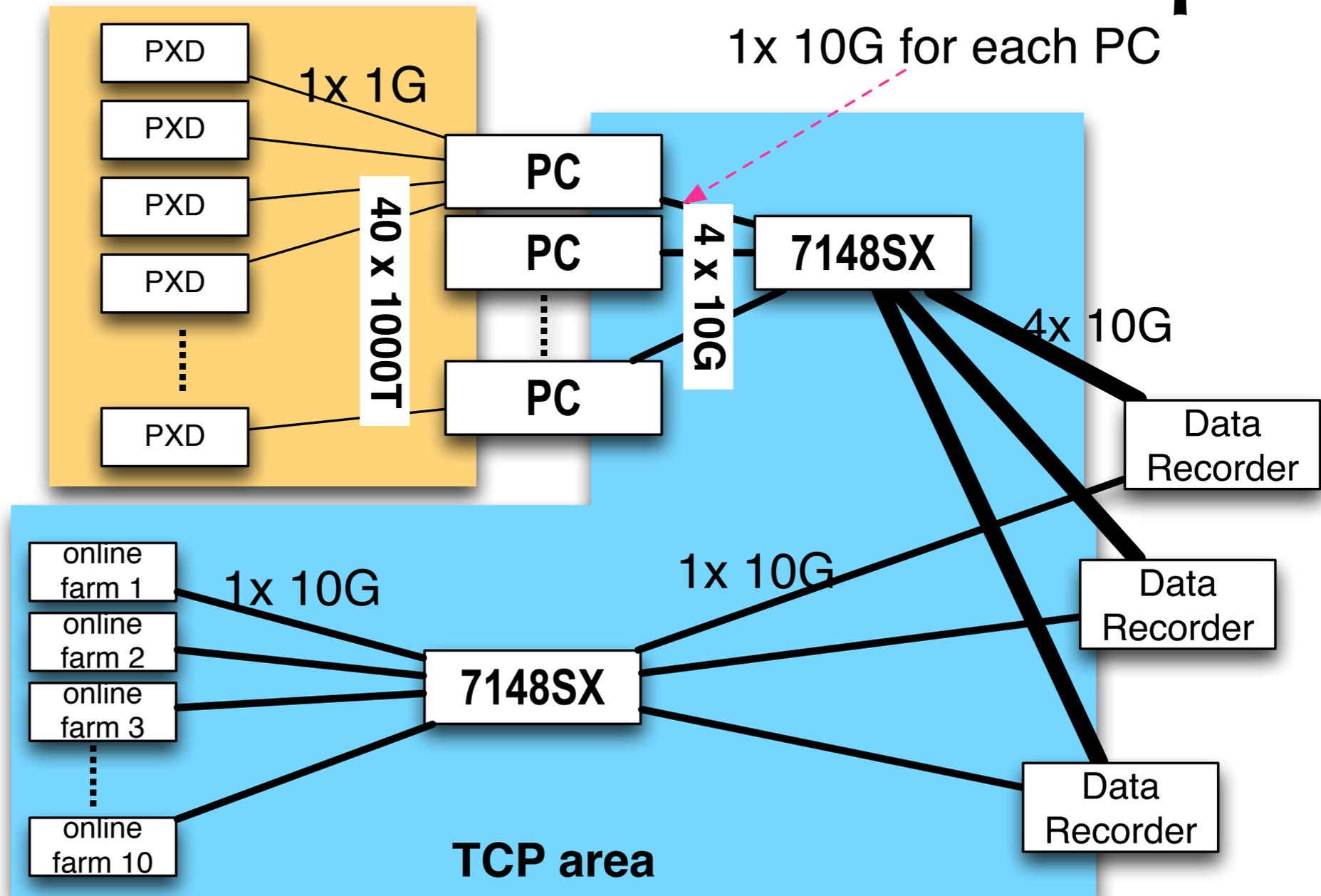
E.B.2

E.B.2

- Receive filtered event fragments from PXD equipment
 - # of PXD output is 40
- Receive half-built event w/o PXD from online farm
 - # of online farm is 2 (beginning) to 10 (design luminosity)
- sends fully-built event to data recorder
 - # of data recorder will be similar to online farm (???)

E.B.2 design

@ last PXD Workshop



15 minutes design

Yesterday,

- Lange-san reported about UDP to TCP conversion by PC
 - also sub event building
 - PC(s) will receive 40 UDP streams from ATCA
 - PC(s) will send data to E.B.2 via TCP streams
 - # of PCs will 10 or more?
- I hope following slides maybe some help to these PC

Difference from E.B.1

- PXD ATCA uses UDP instead of TCP
 - each ATCA will send at least two packets per one event. (right?)
- Event order is not incremental
 - But this order is coincident between output from HLT online farm and output from ATCA
 - Order checker is necessary EBPC of E.B.2

Be careful!

UDP consumes CPU power

- Generally, CPU damage of UDP handling is smaller than that of TCP.
- But recent ethernet LSI has many offload functions to reduce this CPU damage.
 - like a TCP segmentation offload.
- TCP can send multiple short datum in one long packet.
 - this behavior reduces CPU load at the receiver side.
- UDP is not so
- Packet rate effects linearly to the CPU load.

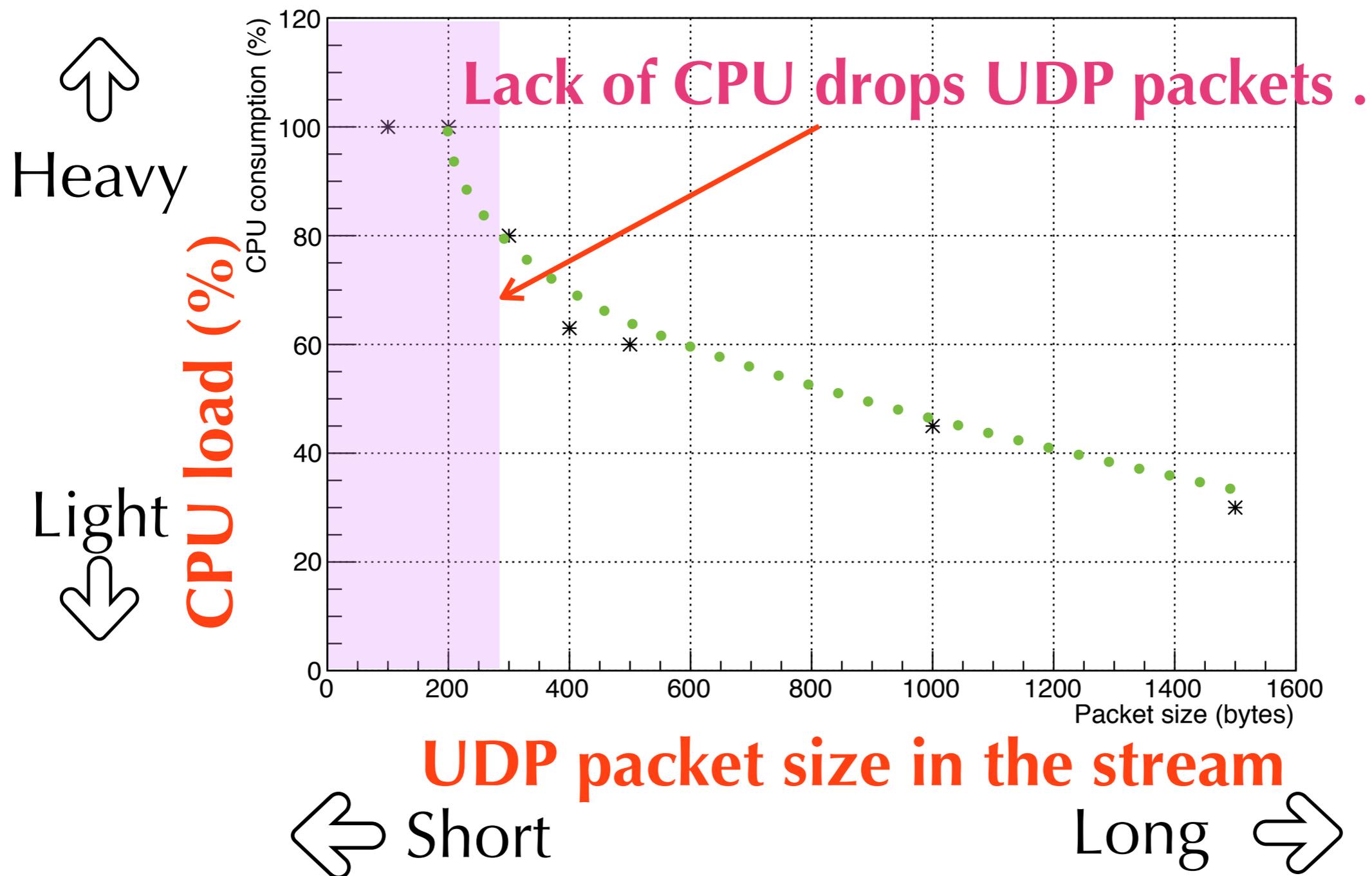
TCP can receive 10Gbps, how about UDP?

- Indeed, TCP can receive 10Gbps, but it requires....
 - TCP segmentation offload (TSO)
 - Large MTU (normally, ethernet will use 1500 byte. 8000 or 9000 bytes for 10G are needed)
 - Large socket buffer (via setsockopt)
- How about UDP?
 - Offload functions are not effective
 - TSO and GRO are for TCP, will not help UDP.
 - Large MTU will not help for short message stream
 - In the case of PXD,
 - two packets per event
 - Packet rate will be higher than that of COPPER
- How about packet size? that is very important!

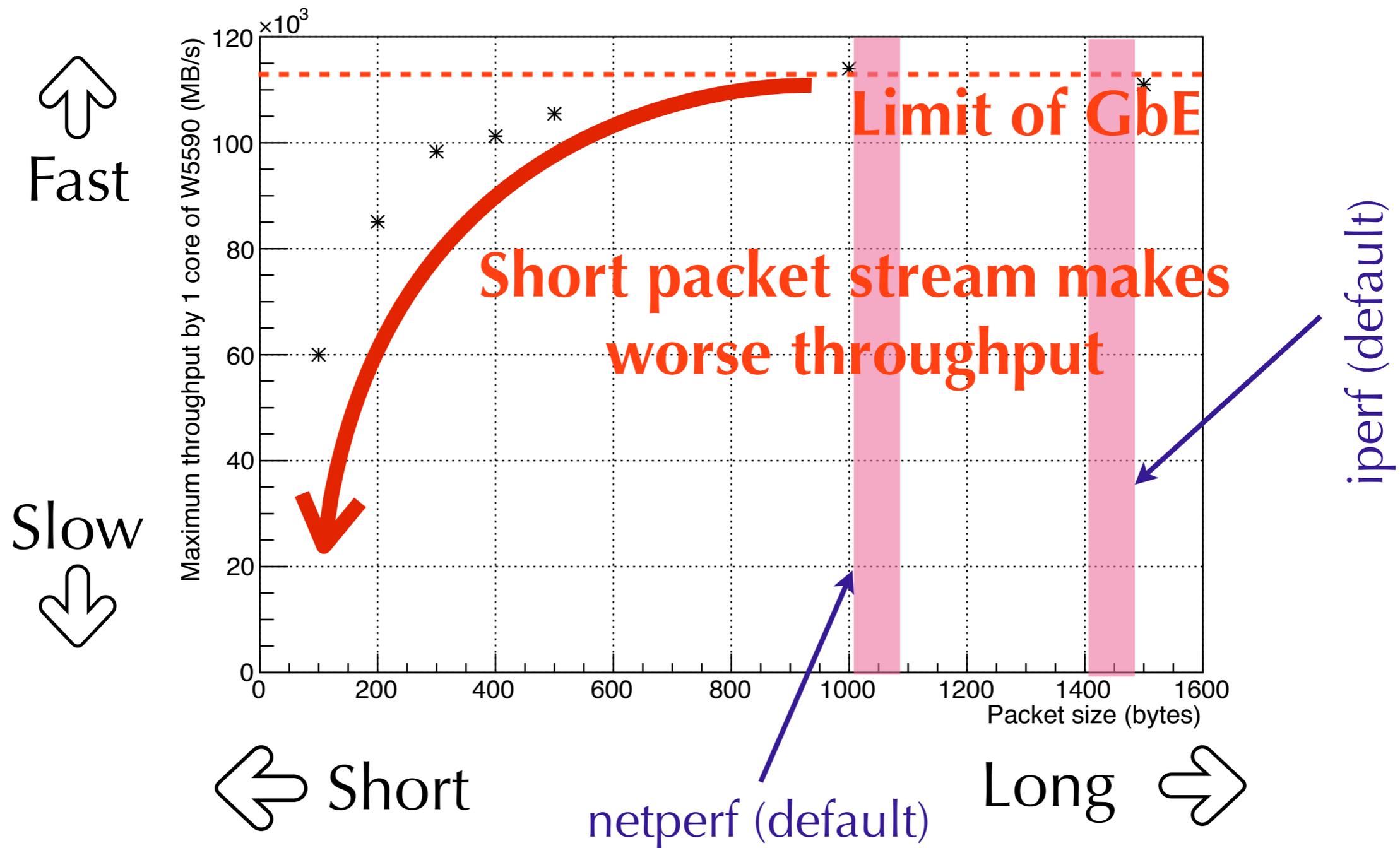
CPU damage test

- Tester
 - S: Sends UDP packets without any software limitation
 - R: Receiver reports frequency and drop
- 100byte per packet, receive rate is about 600kHz
 - UDP receiver consumes 100% of 1 core of W5590 (Xeon 3.3G)
 - very often packet loss
- 500B per packet, 211kHz
 - 60 % of W5590
- 1000B per packet, 114kHz
 - 45 %
- 1500B per packet, 74kHz
 - 30 %

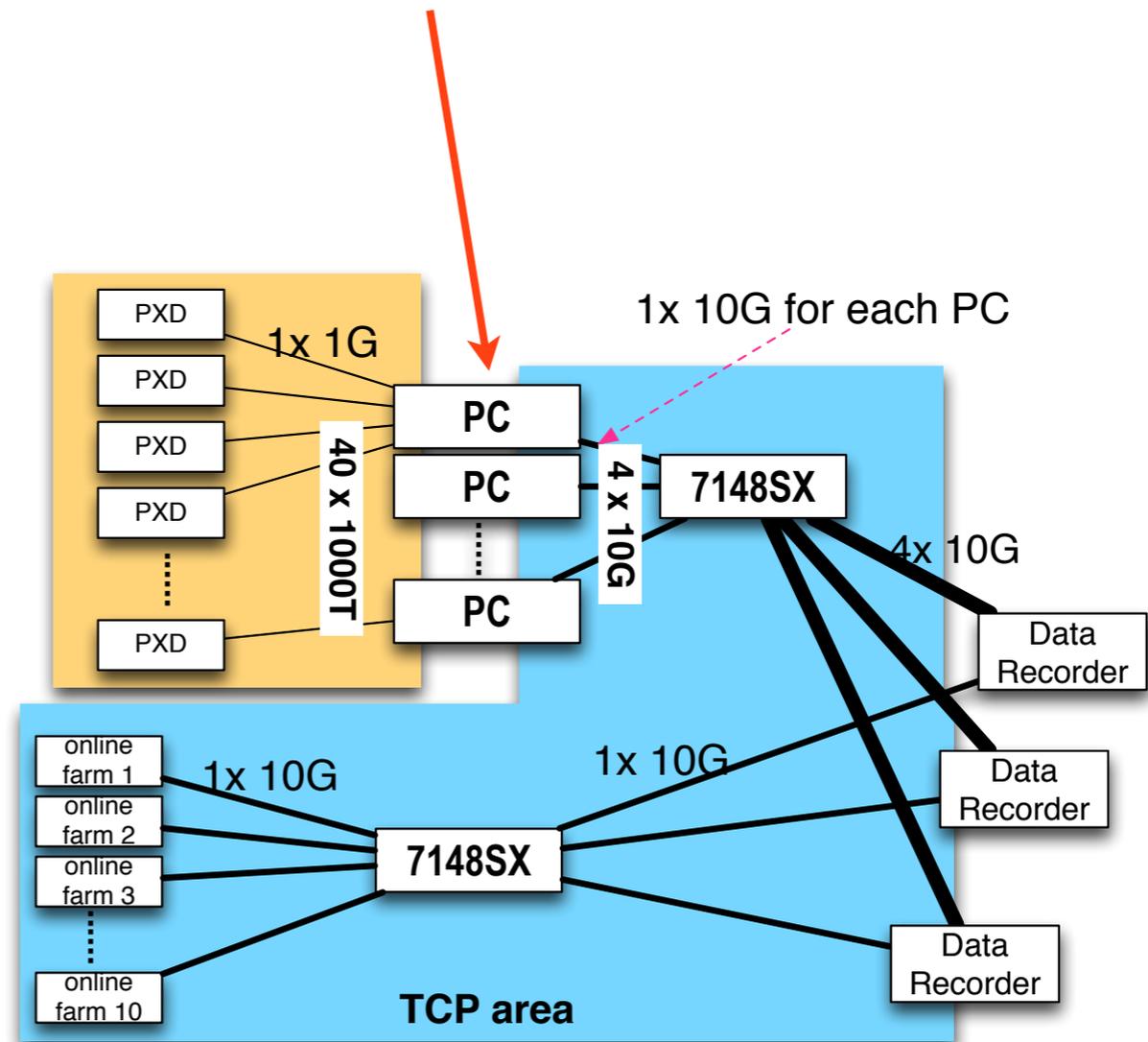
Like this



Achievable speed of UDP stream



If packet size is shorter than 500byte,
8 or more cores are needed to receive
8Gbps, around here.

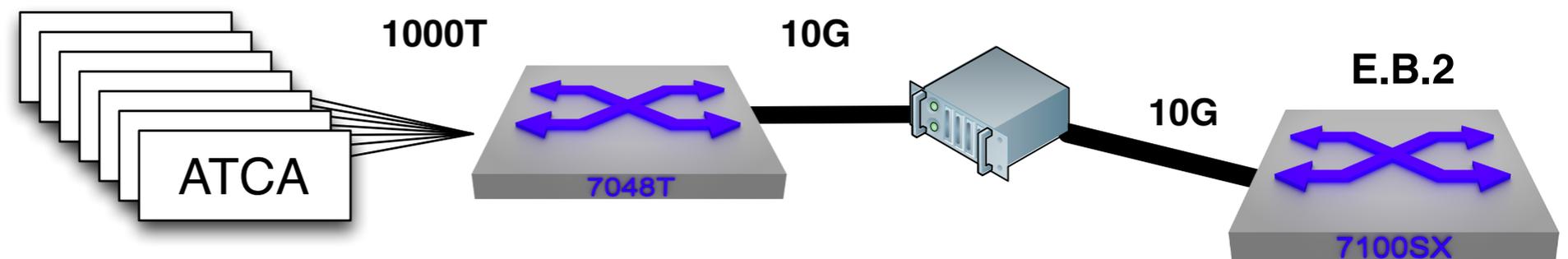


How connect from
ATCA and PCs?

Switch from ATCA to PC

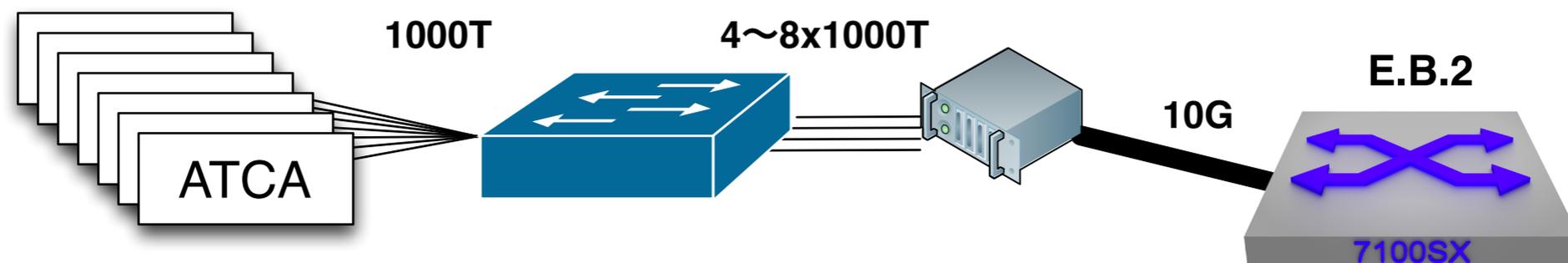
(1GT => 10G)

Vendor	Model	Price (yen)	# of 10G
ARISTA	7050T-A	1M	4
DELL	Powerconnect 8024	1M	24
DELL	Powerconnect 7024	650k	2
DELL	Powerconnect 5224	240k	2
HP	Procurve	1.5M	
cisco	Catalyst 2960S-24TD	270k	2



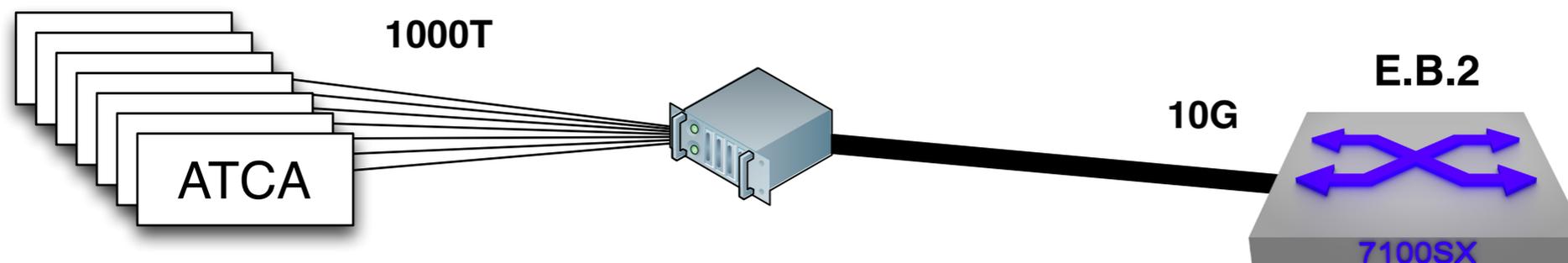
cheaper option (w/o 10G)

- Use “bonding” of 1000T instead of 10G links
 - IEEE 802.3ad standards
- Planex SW-0224G2 (24port) => 50k
 - max 4 physical links per logical link
- Allied Telesis switches (max 8 physical per logical) => 240k
- But do not expect “8 x 1000T” will achieve 8Gbps
 - Which physical link should be used to transmit packet?
- This option is not so reliable because,
 - bonding behavior in cheap switch is not clear
 - survey is necessary



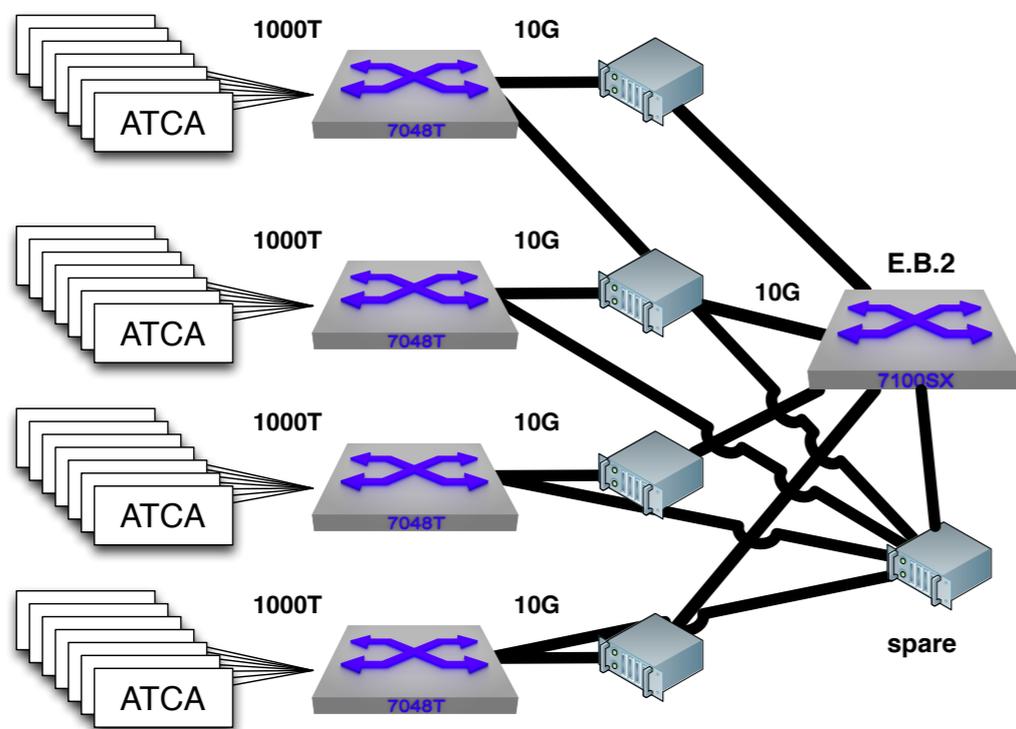
More cheaper option

- Direct connection from ATCA to PC
 - One PC can equip about 10 of GbE by Quad port Ethernet card
 - Looks like the Belle I event builder
 - cost per GbE port is about 5k yen
- It is better to keep throughput < 10Gbps per PC
 - More than 10 ports, maybe danger because of CPU consumption
- If 10 PCs are assigned, only 4 ports per PC are sufficient



Problem of direct connection

- PC hardware reliability
 - If some of PC has broken, whole DAQ can't take data any more.
 - Danger to order re-cabling to shift crew.
 - At the end of Belle I experiment, experiment shift can't find "START" button on the operation GUI.
- 1st option (using switches and 10Glinks) is better, I think.



Spare PC must equip many 10GbE, but only one is used at a time.

7048T has 4 of 10G links, so 3 spares can be connected like this.

DELL or cisco, only one spare can exist.

Summary of E.B.2

- UDP-TCP converter PCs will be assigned in front of E.B.2.
 - They will run similar software to ROPCs
- Data recorder (or other PC in front of data recorder) will run similar software to EBPCs.
- We have to clear following parameters to make prototype
 - Expected UDP packet size
 - with or without hit information
 - Expected throughput per link
- Study of DELL powerconnect and cisco cheap switch maybe necessary

That's all,
but stability test is ongoing.

Modification for local run

- Configuration data for each local run group.
 - EBPC
 - IP address and port number of ROPCs that should be connected
 - ROPC
 - IP address and port number of COPPERs
 - In the case of BelleII, ROPC *may* receive multiple detectors
- Something to execute “local-run script”
 - like a “Hi NIM pulser, lets begin several Hz periodical trigger”
 - should be implemented in E.B.? (I don't think so)

Multiple IPs on single ROPC

- Linux can have multiple IP address for each ethernet, using name space ethX:N (N is natural number)
 - These addresses can be in the same subnet.
- If ROPC1 handles some of CDC and some of ECL,
 - 172.22.106/24 is assigned for Global, 172.22.107.11/24 for CDC, 172.22.108.11/24 for ECL, and so on
 - eth0 => 172.22.106.11 for Global data taking
eth0:1 => 172.22.107.11 for CDC localrun,
eth0:2 => 172.22.108.11 for ECL localrun,