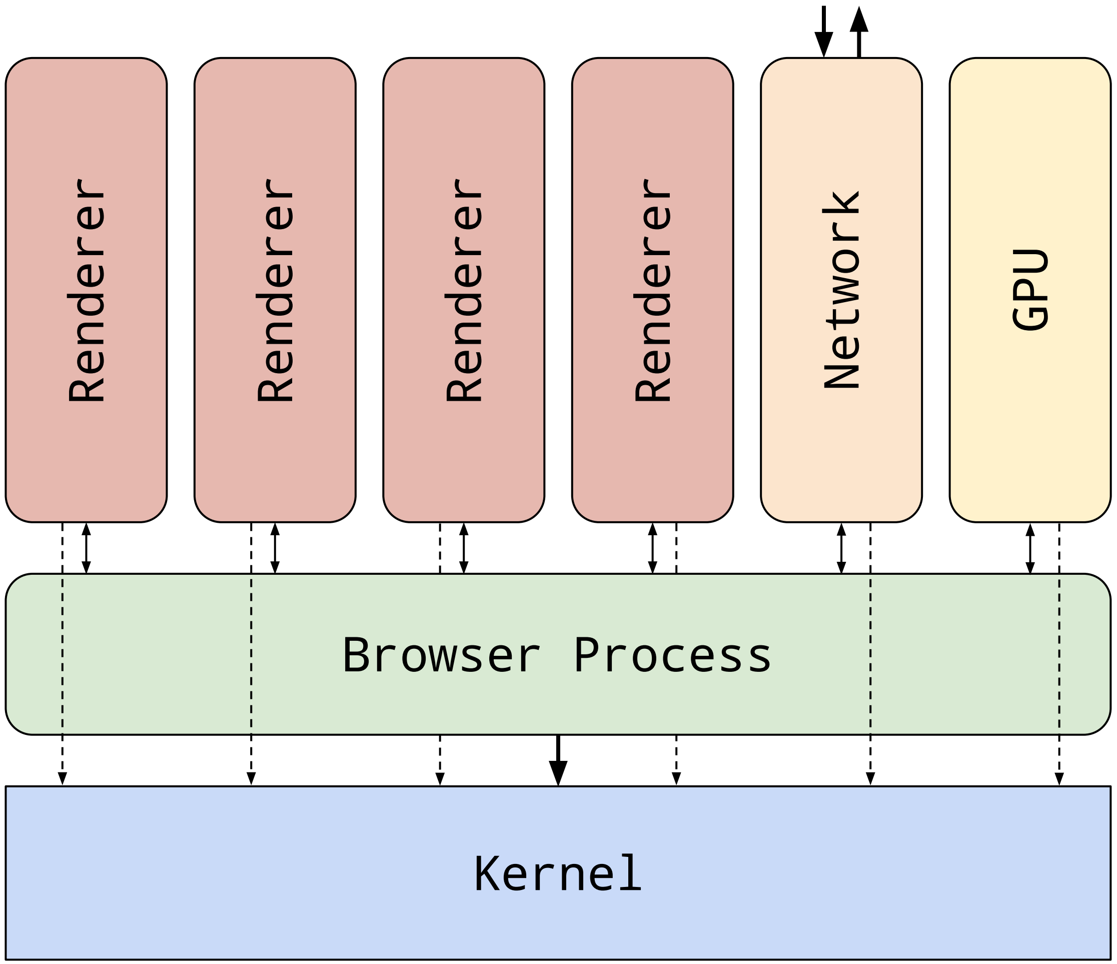
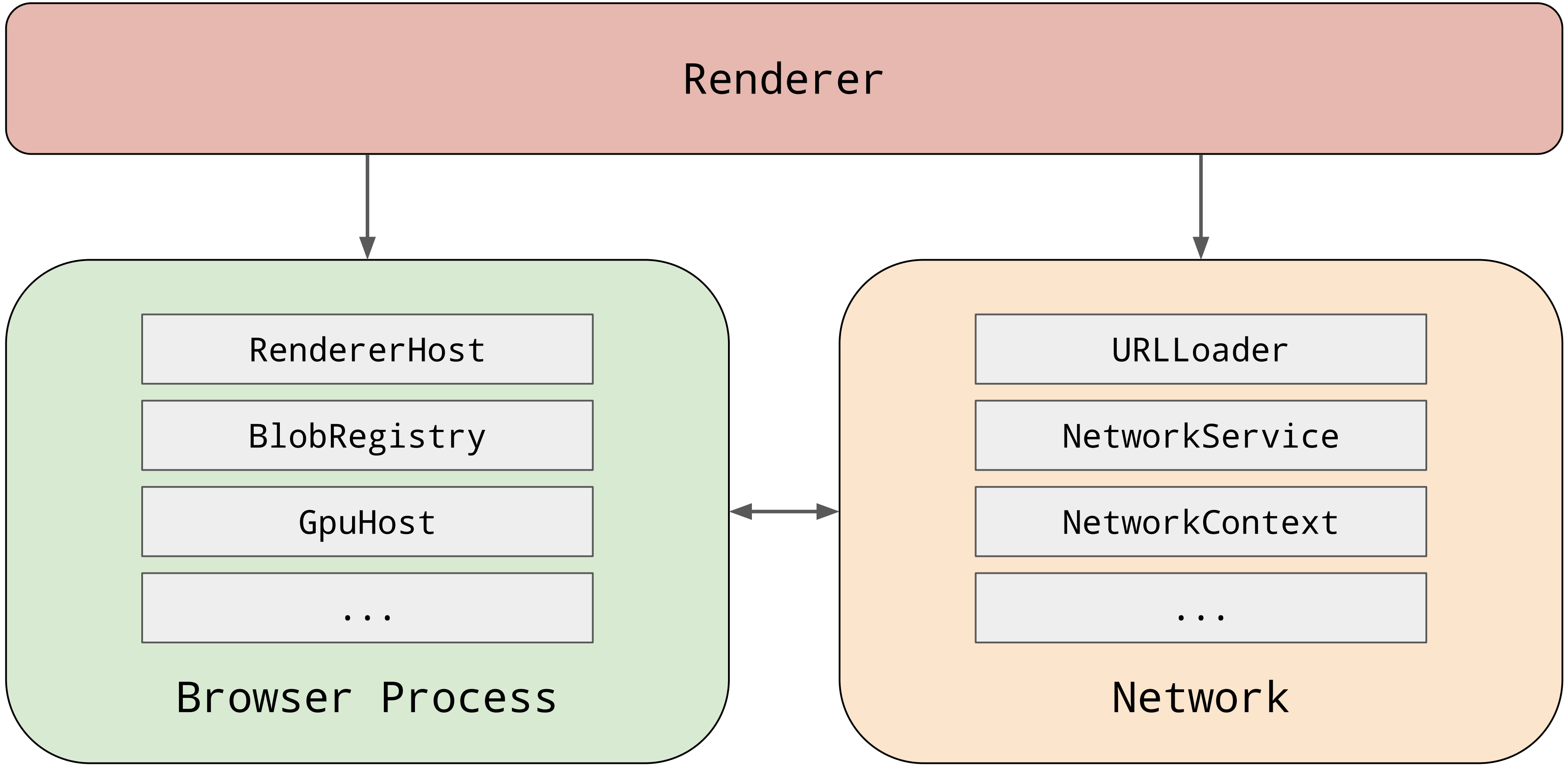
Breaking the Chrome Sandbox with Mojo

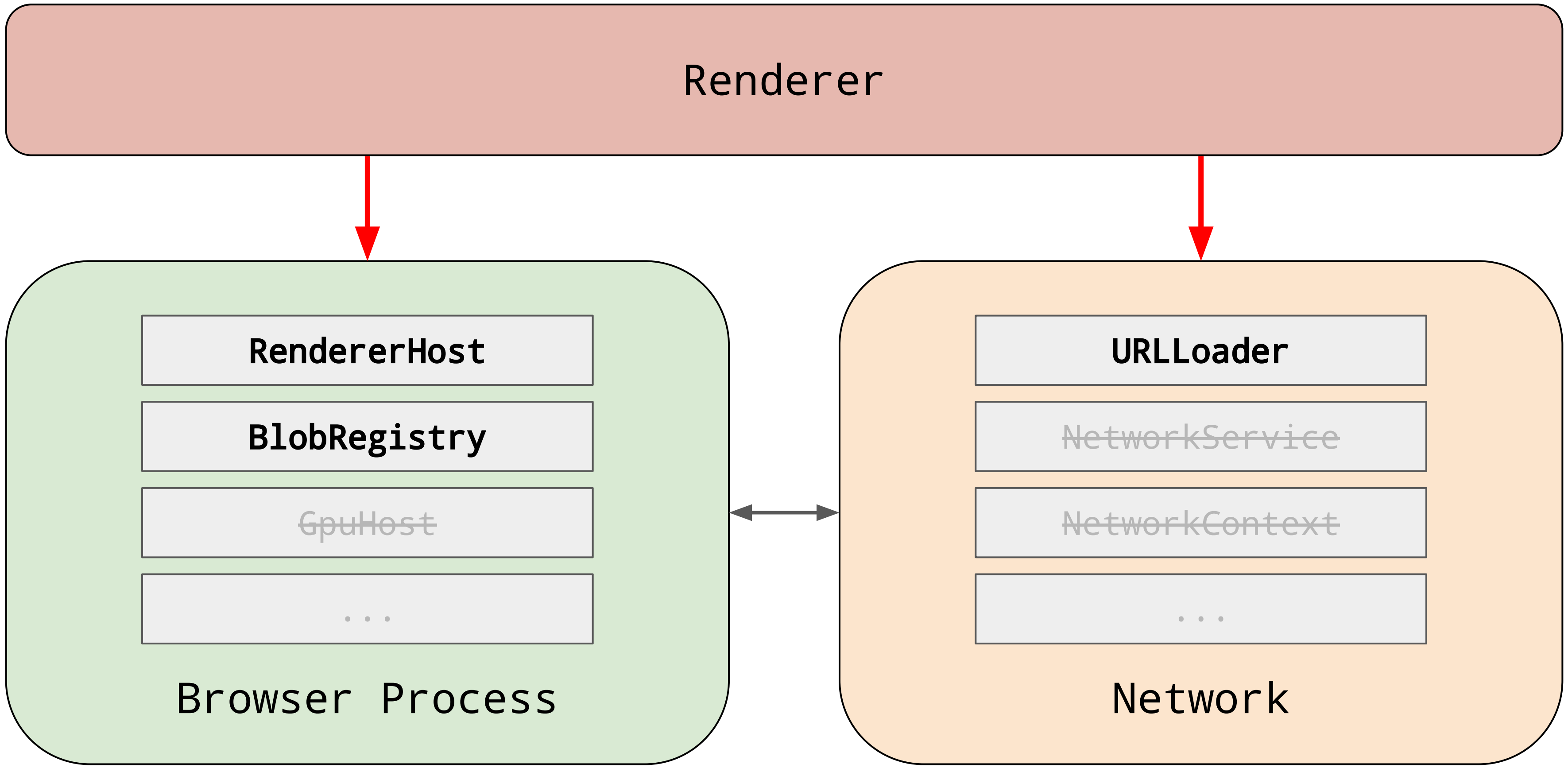
_tsuro@, 2022

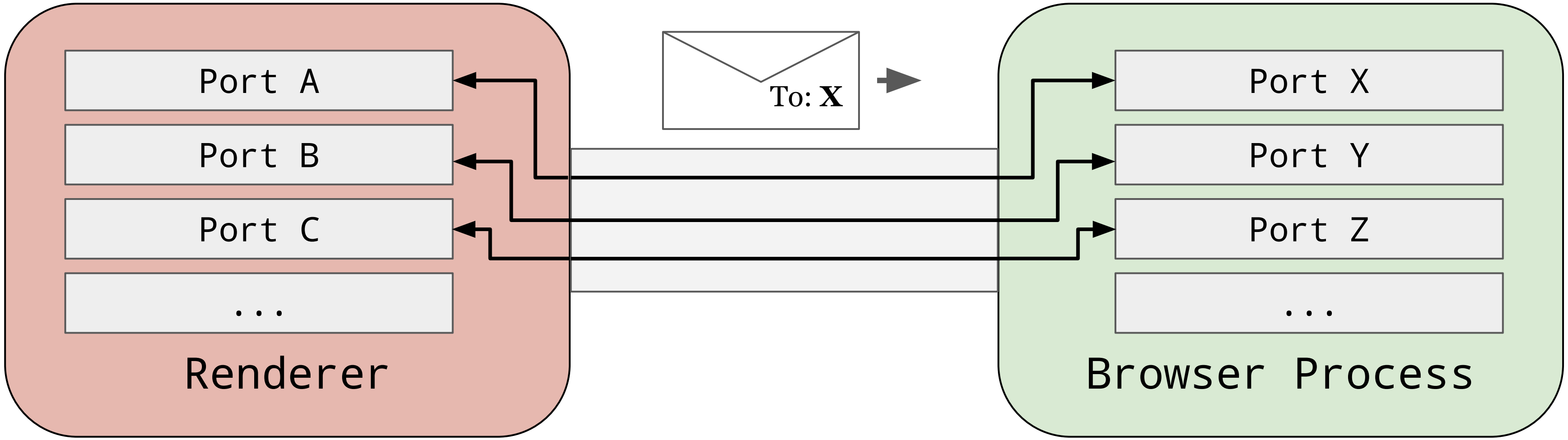
```
 _________________________________
| Calculator         | - | O | X ||
|---------------------------------|
|  _____________________________  |
| |                             | |
| |                       1337  | |
| |_____________________________| |
|                                 |
|  _____ _____ _____ _____ _____  |
| |     |     |     |     |     | |
| | MC  | MR  | MS  | M+  | M-  | |
| |_____ _____ _____ _____ _____| |
| |     |     |     |     |     | |
| | <-- | CE  |  C  | +/- | sqr | |
| |_____|_____|_____|_____|_____| |
| |     |     |     |     |     | |
| |  7  |  8  |  9  |  /  |  %  | |
| |_____|_____|_____|_____|_____| |
| |     |     |     |     |     | |
| |  4  |  5  |  6  |  *  | 1/x | |
| |_____|_____|_____|_____|_____| |
| |     |     |     |     |     | |
| |  1  |  2  |  3  |  -  |     | |
| |_____|_____|_____|_____|  _  | |
| |           |     |     |  _  | |
| |     0     |  -  |  +  |     | |
| |___________|_____|_____|_____| |
|_________________________________|
```

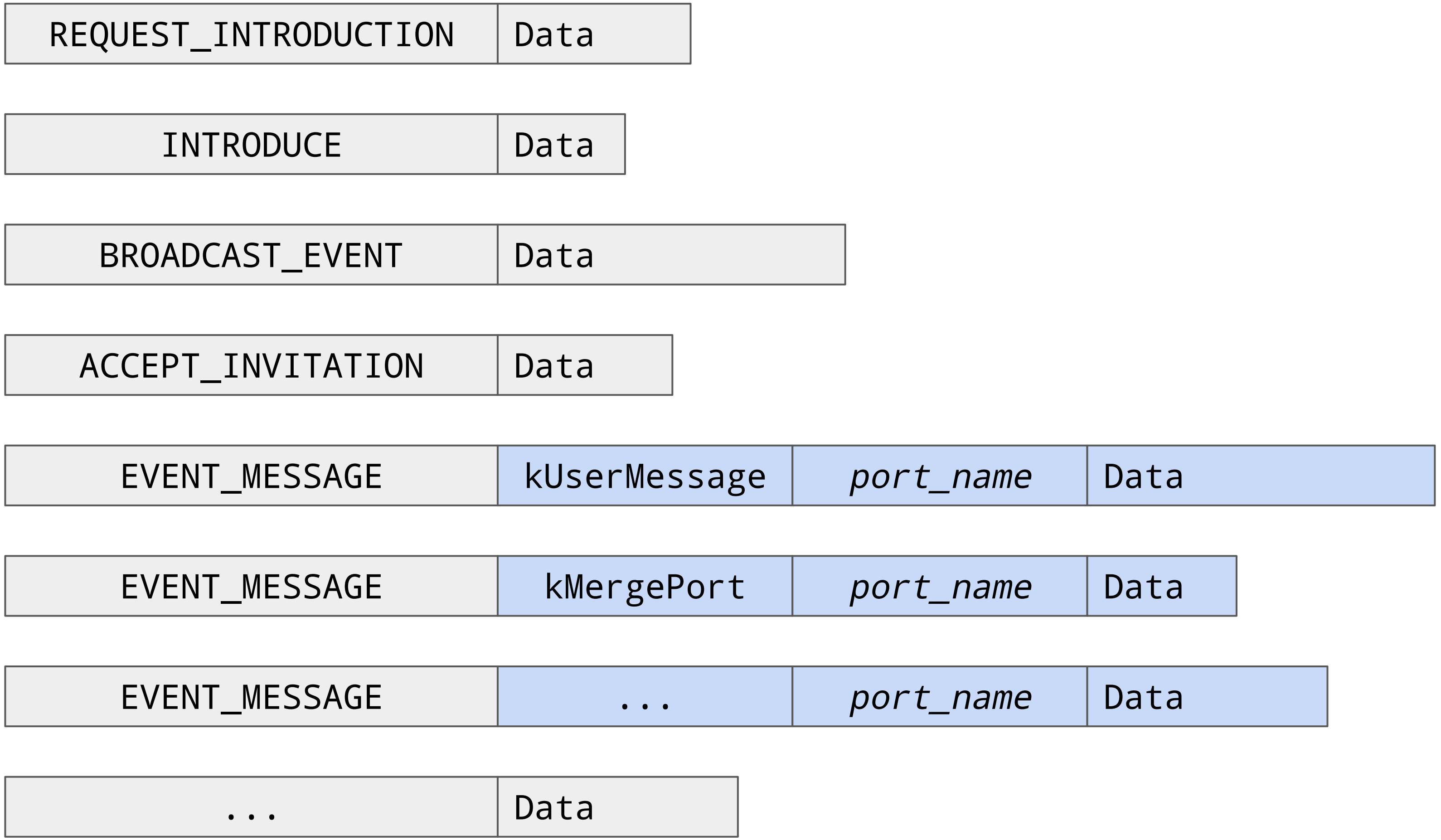
No memory was corrupted in the making of this presentation

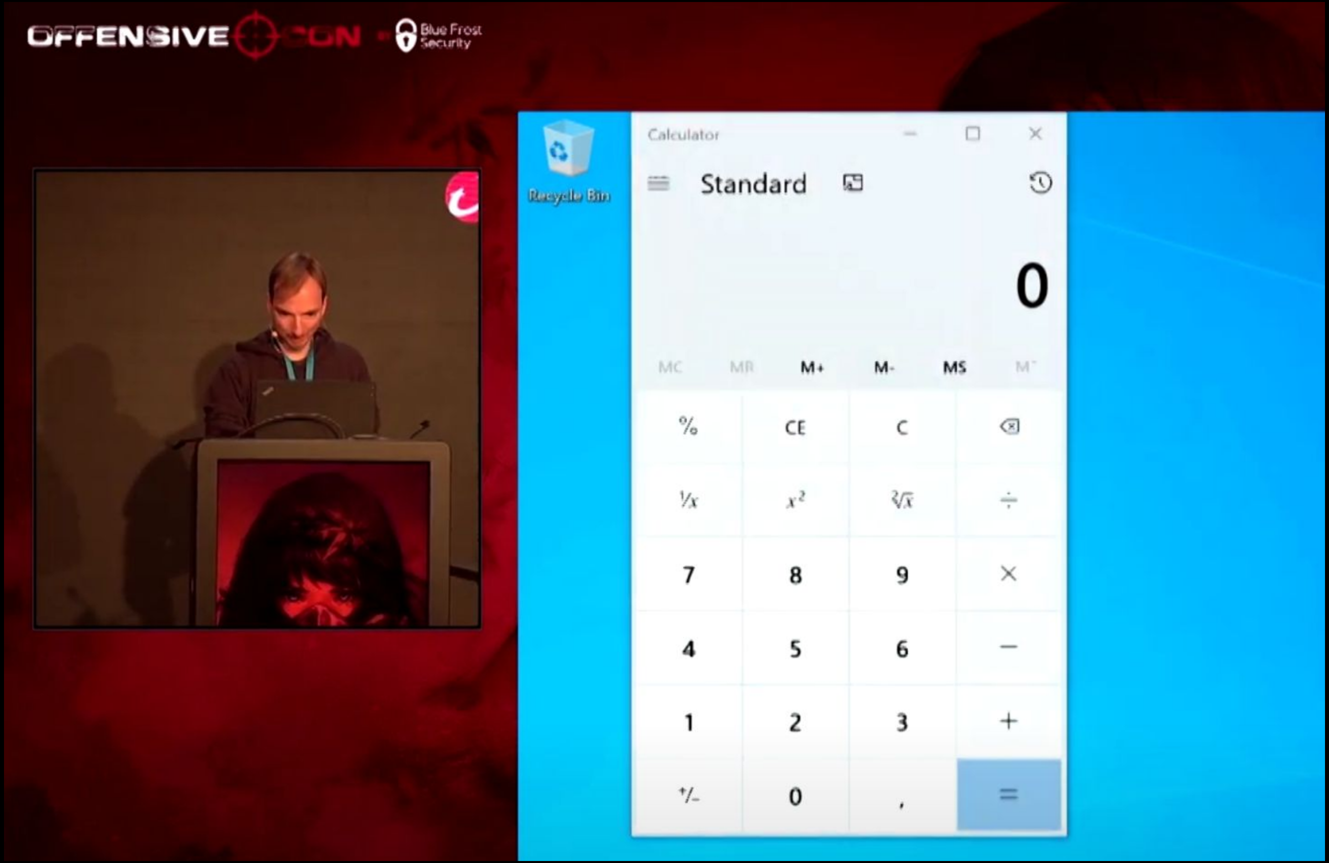

Renderer
Renderer
Renderer
Renderer
Network
GPU
Browser Process
Kernel


Renderer
RendererHost
BlobRegistry
GpuHost
...
Browser Process
URLLoader
NetworkService
NetworkContext
...
Network


Renderer
RendererHost
BlobRegistry
~~GpuHost~~
...
Browser Process
URLLoader
~~NetworkService~~
~~NetworkContext~~
...
Network

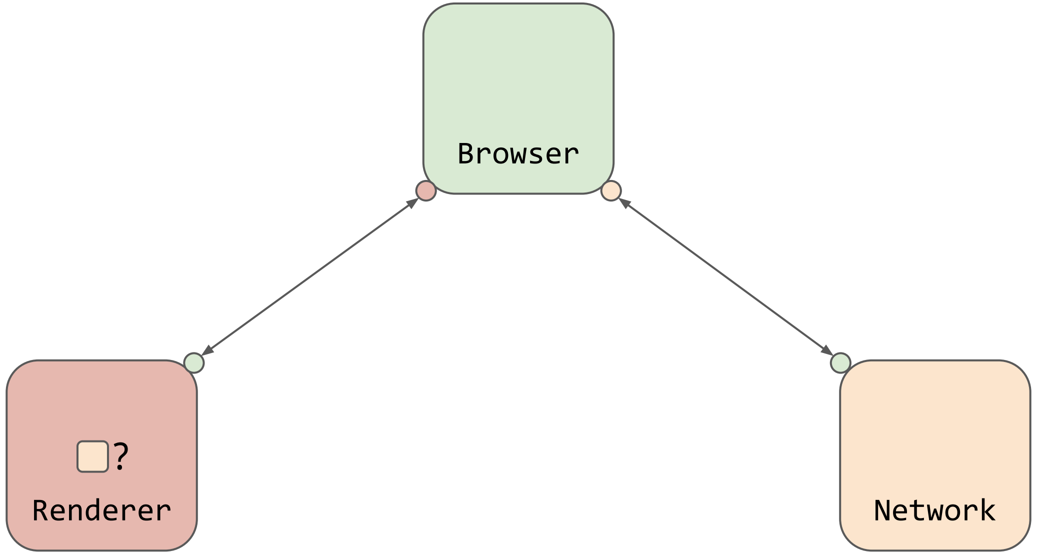

Port A
Port B
Port C
...
Renderer
To: X
Port X
Port Y
Port Z
...
Browser Process

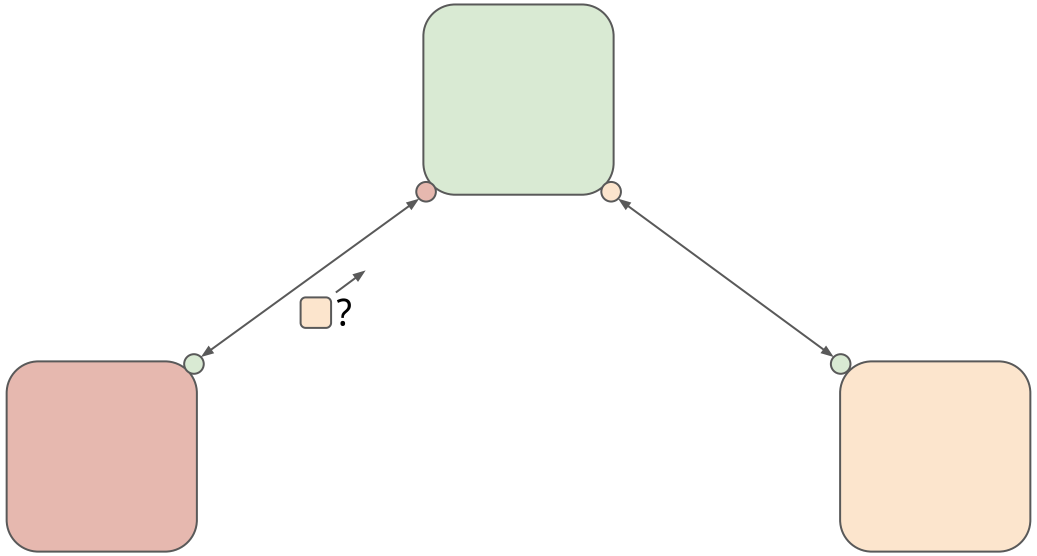

REQUEST_INTRODUCTION
Data
INTRODUCE
Data
BROADCAST_EVENT
Data
ACCEPT_INVITATION
Data
EVENT_MESSAGE
kUserMessage
port_name
Data
EVENT_MESSAGE
kMergePort
port_name
Data
EVENT_MESSAGE
...
port_name
Data
...
Data

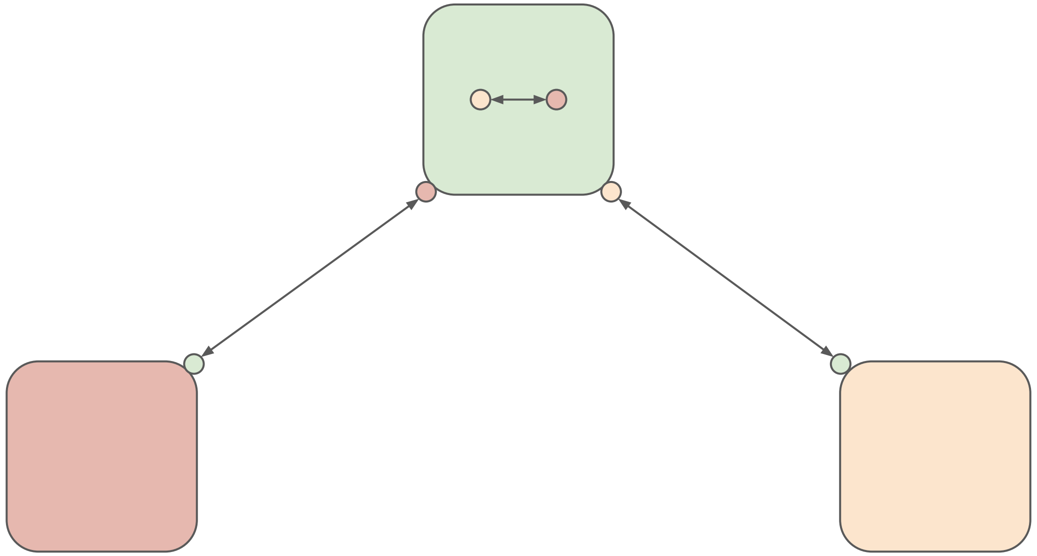


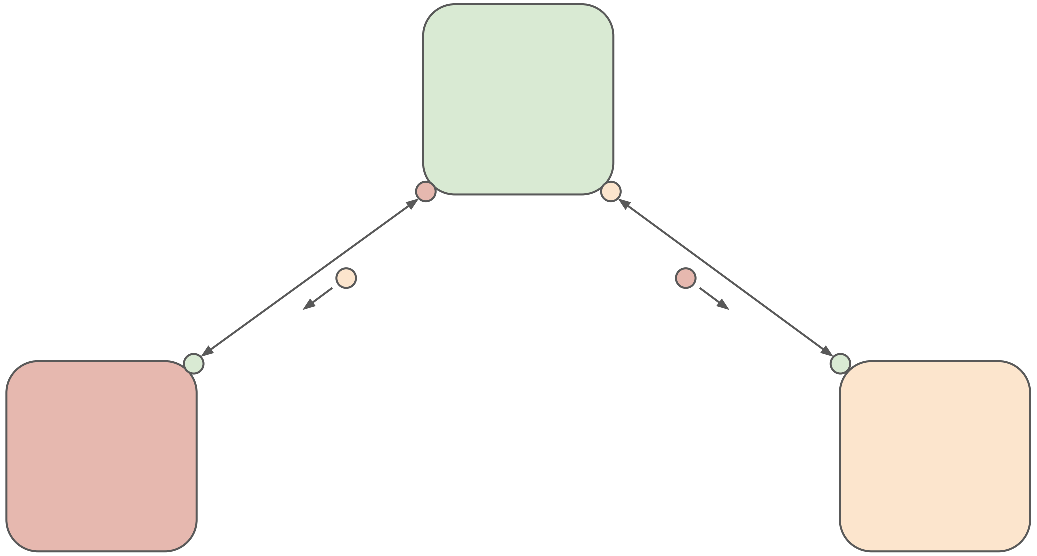
OffensiveCon 2020 - Popping Calc with Hardware Vulnerabilities

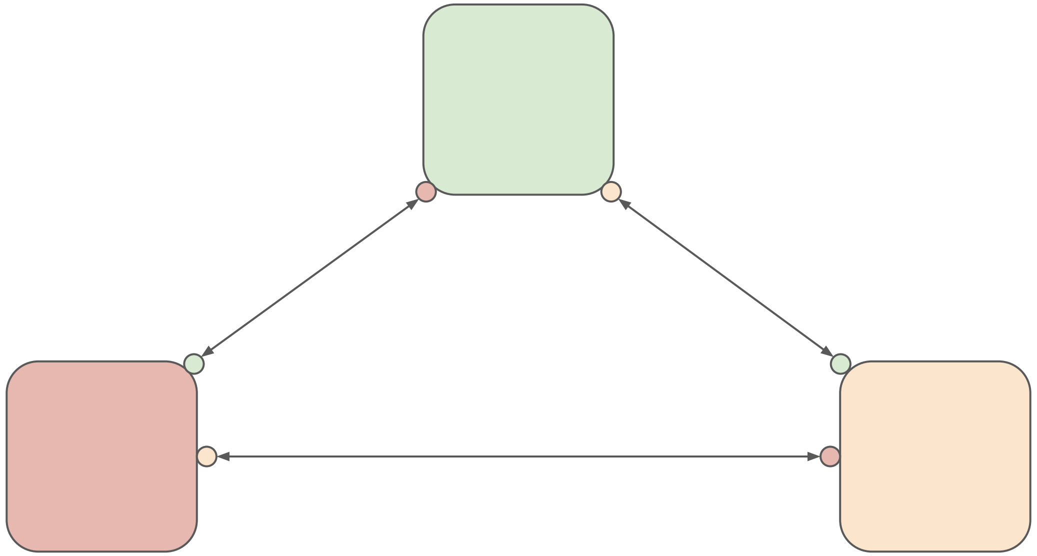
Leaking ports == bad

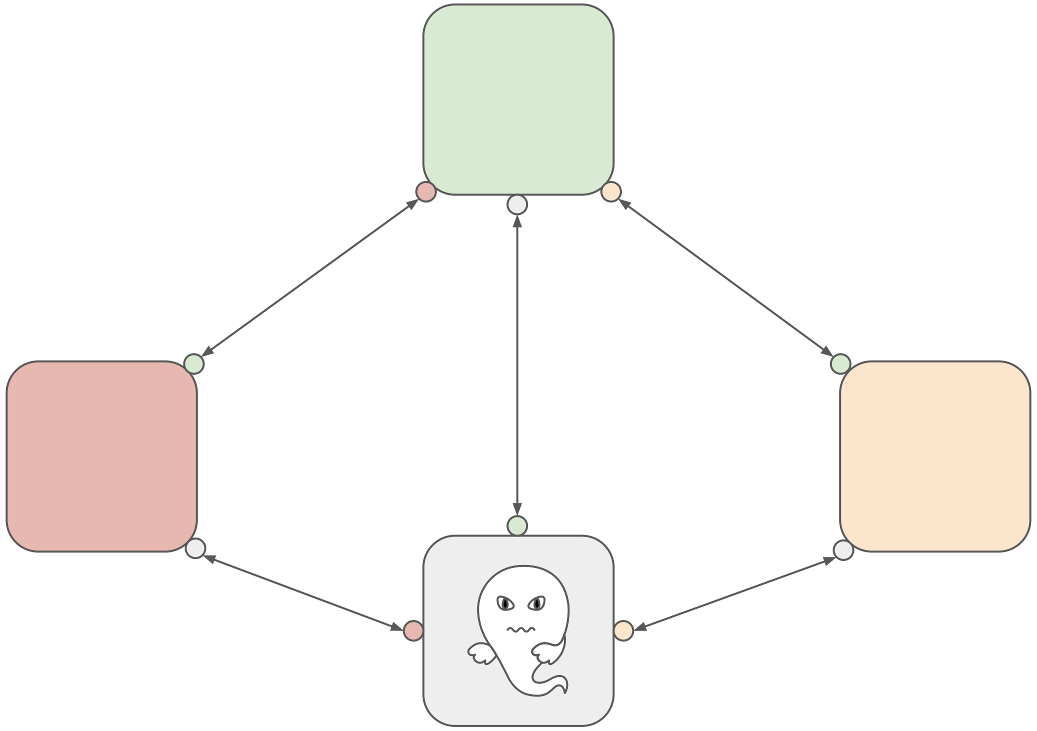

Browser
?
Renderer
Network

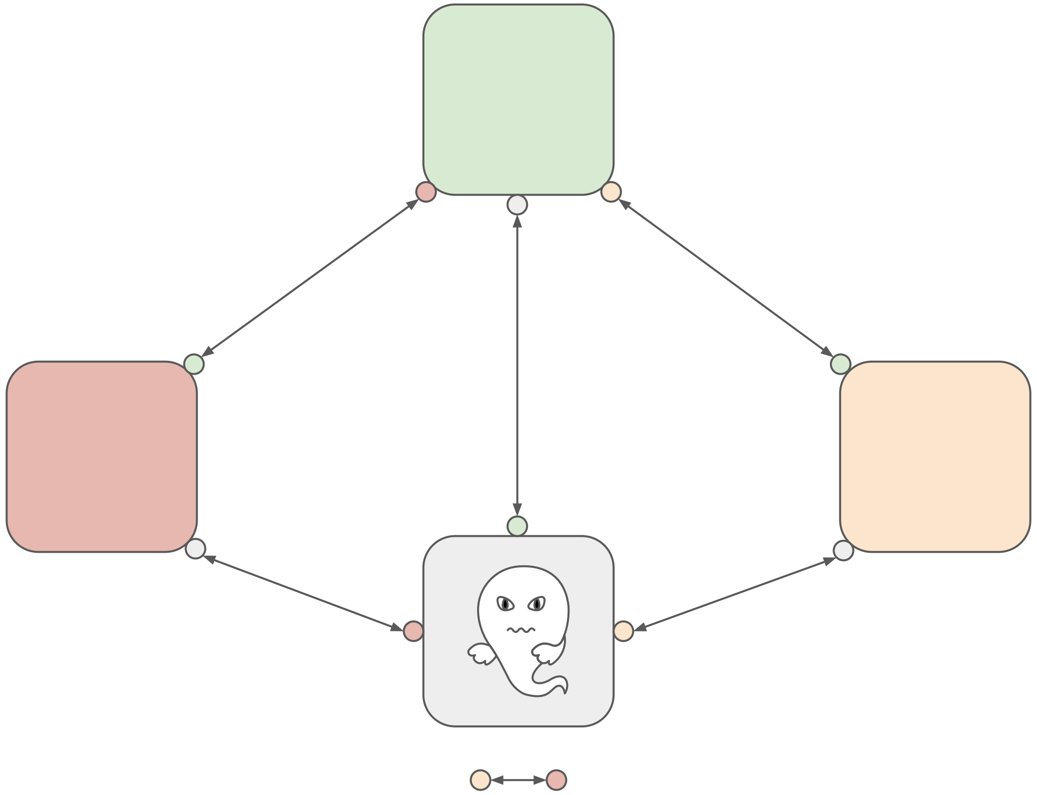


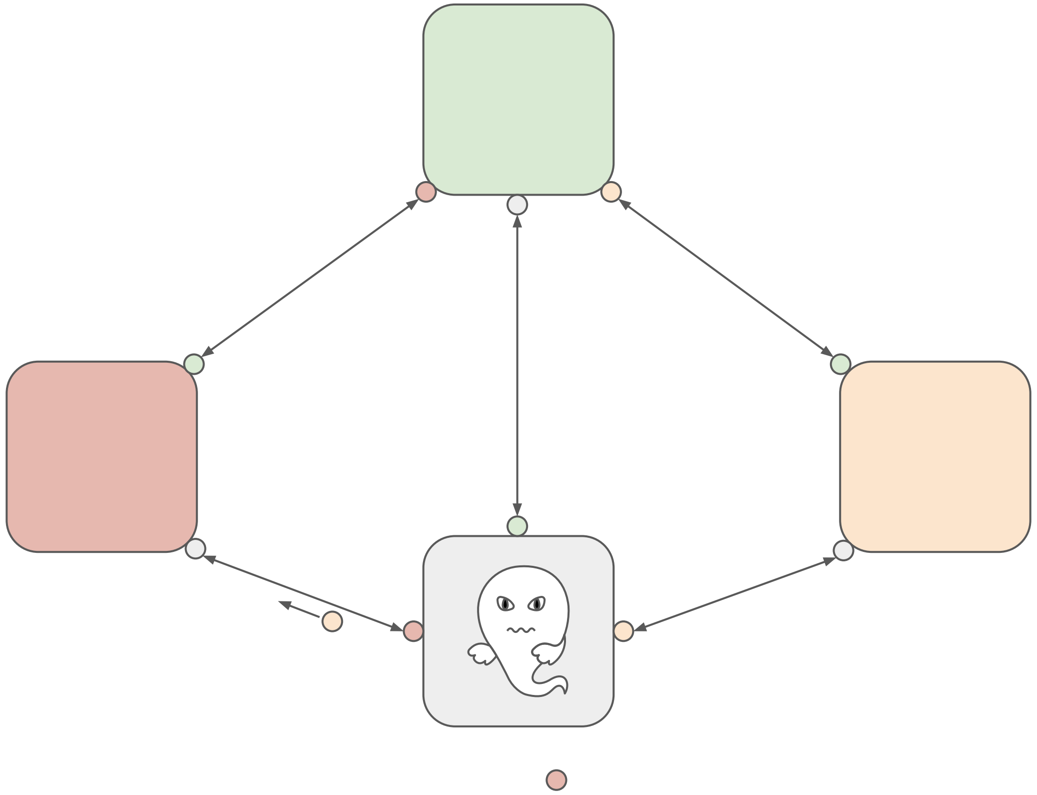


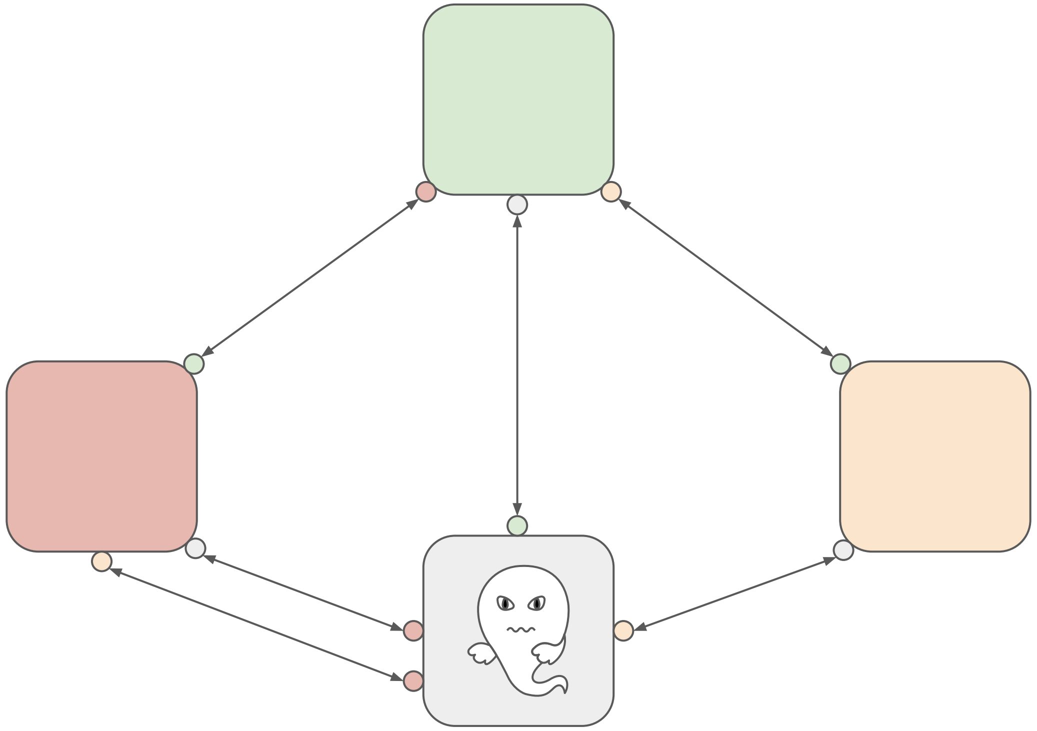


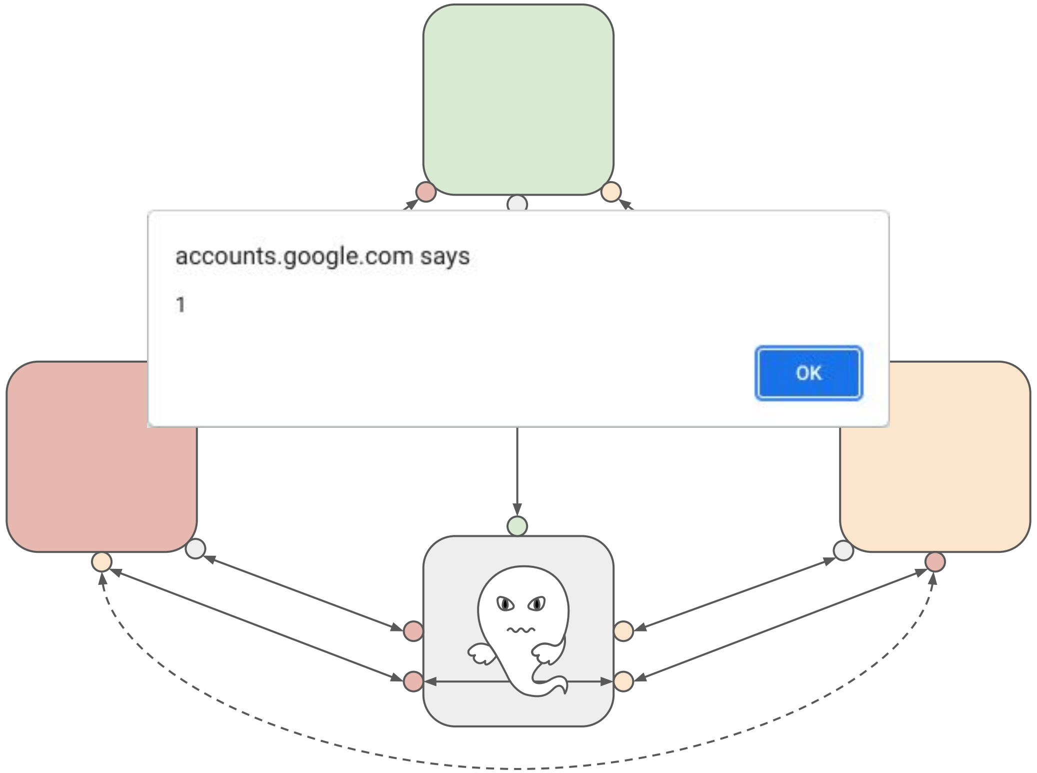






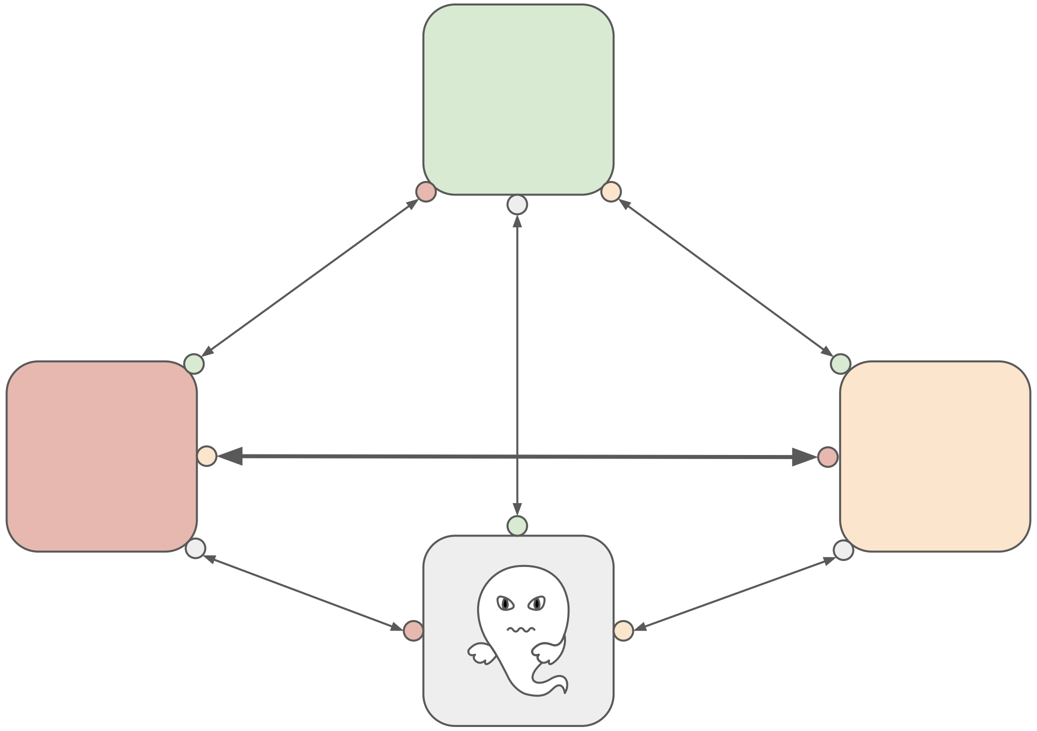


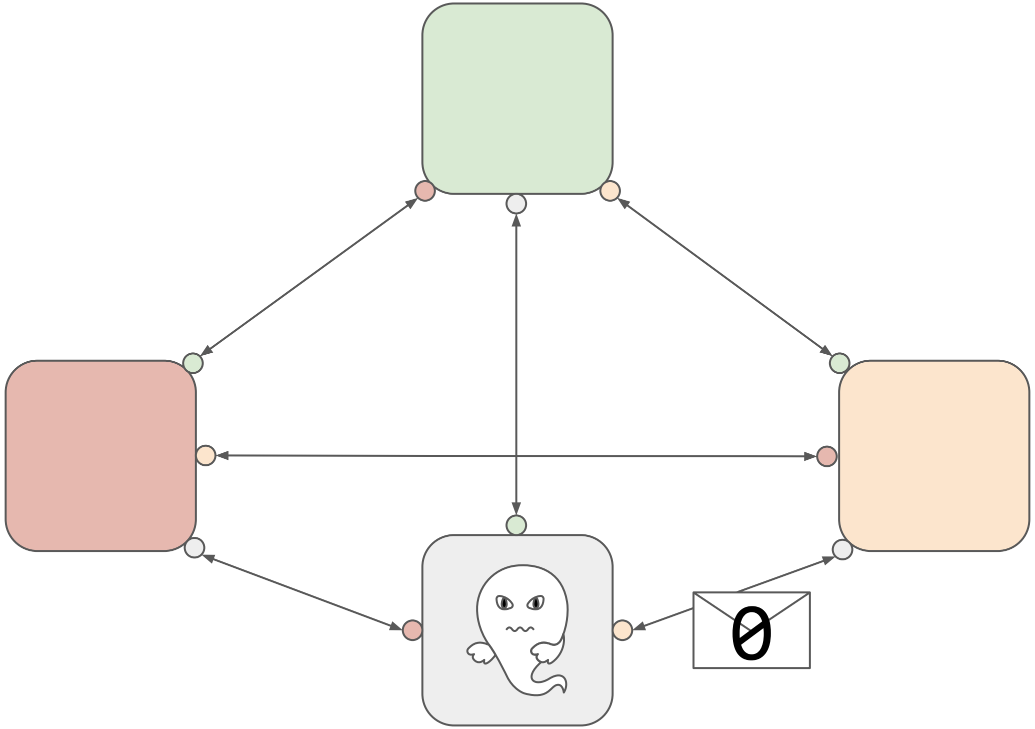


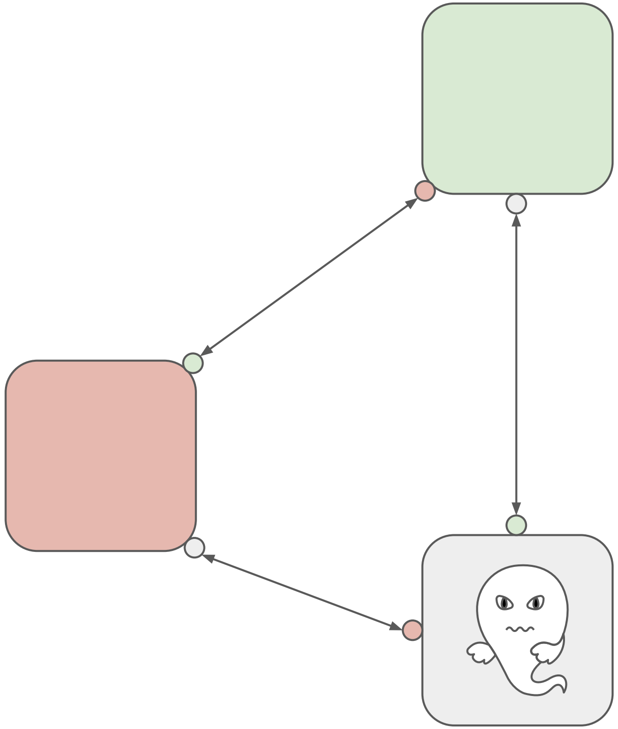


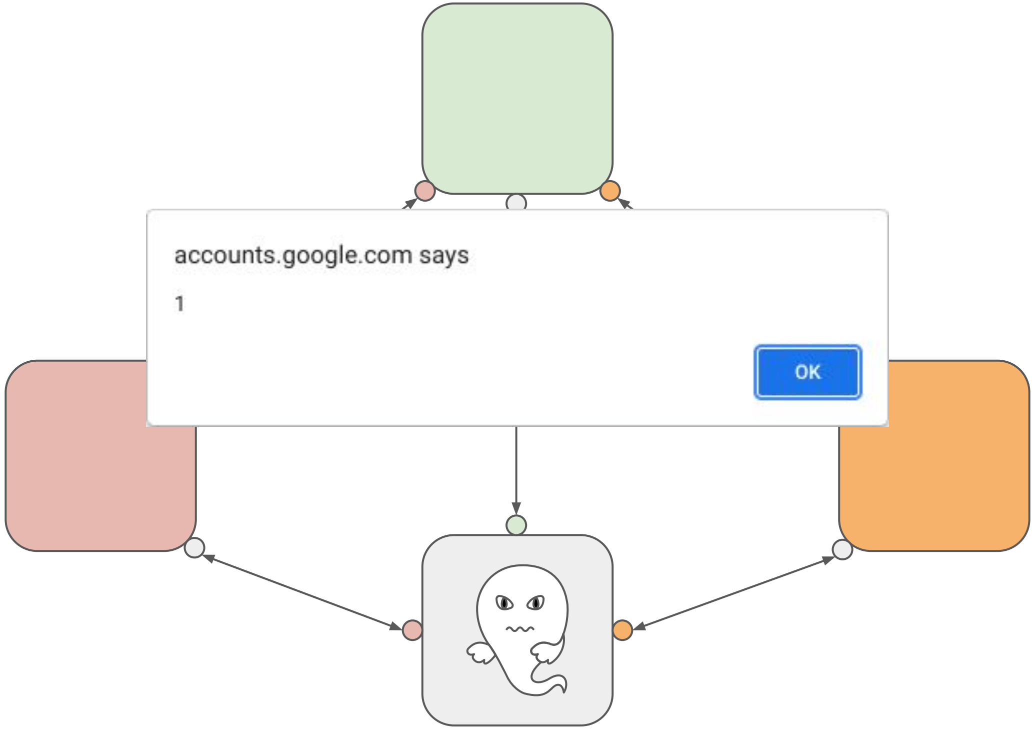
accounts.google.com says

1

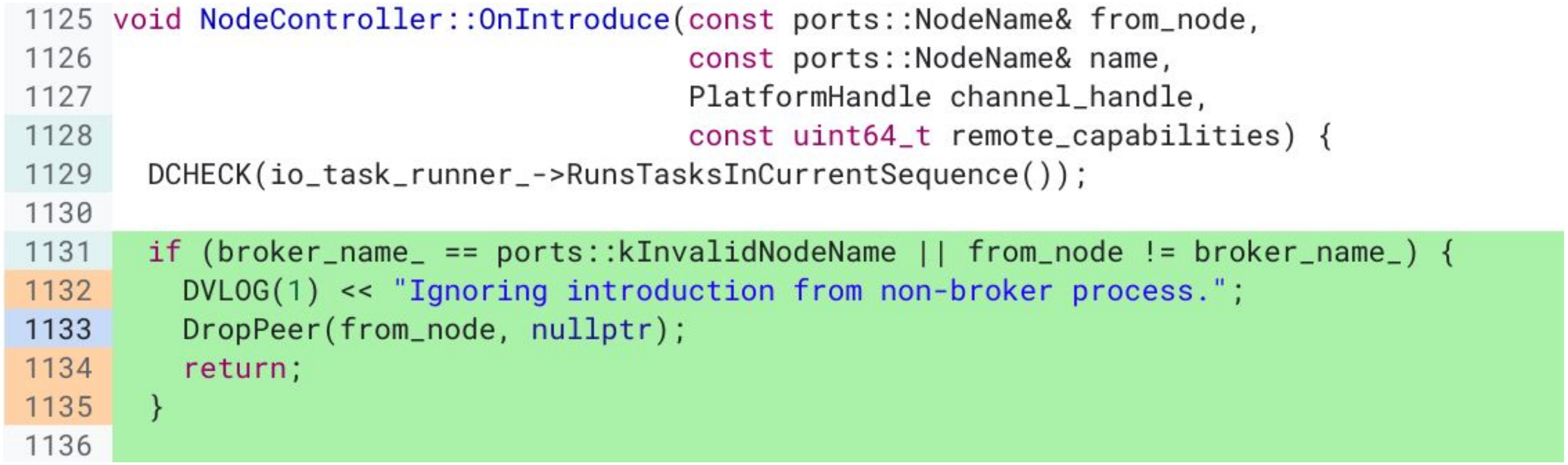



0

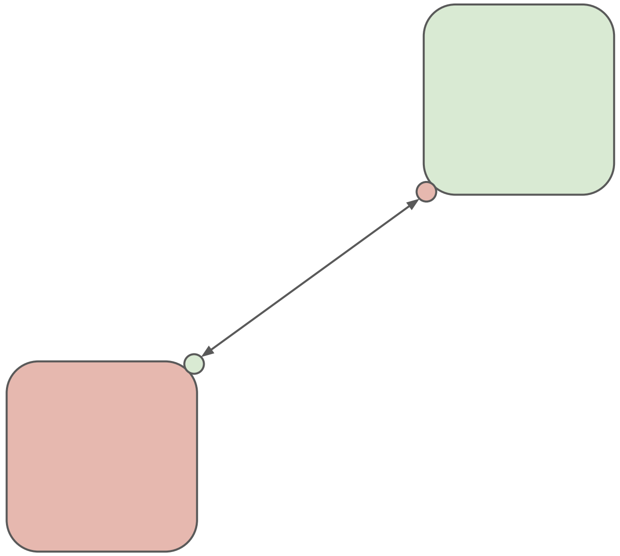


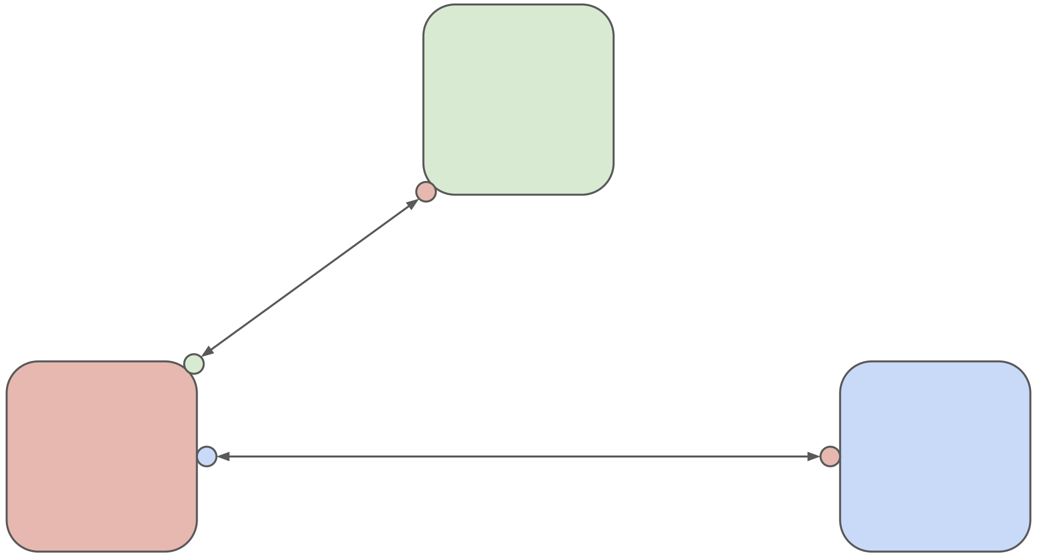

accounts.google.com says

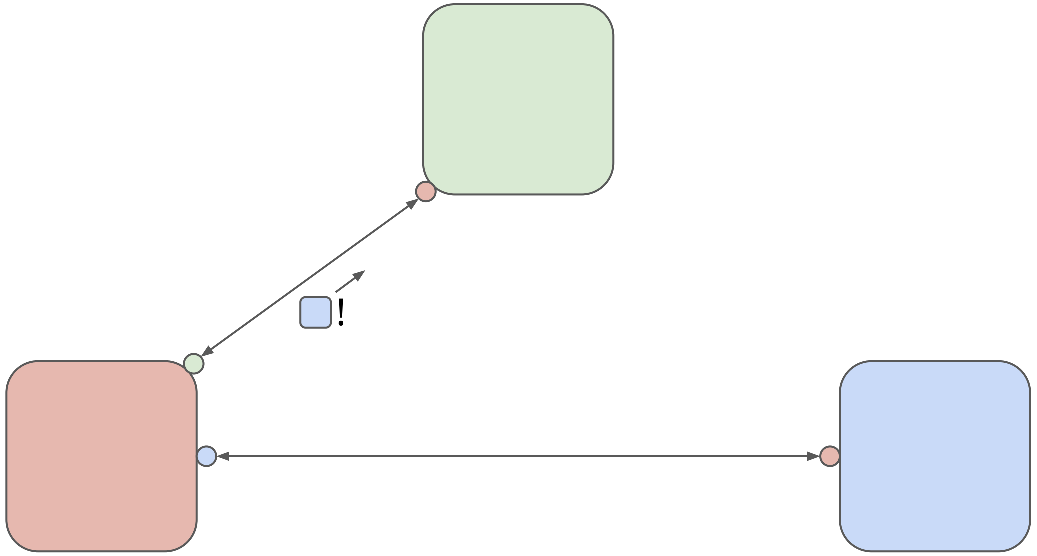
1

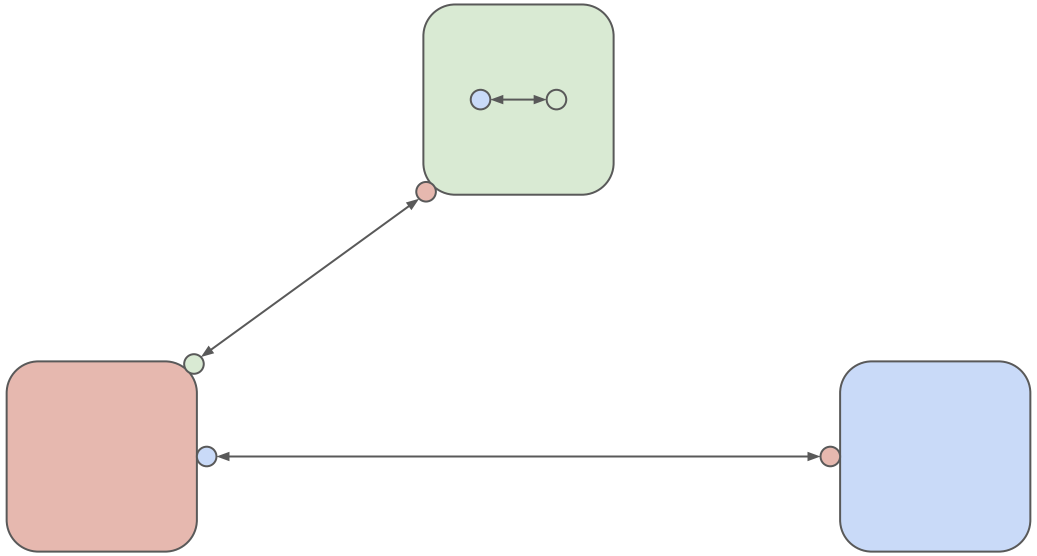


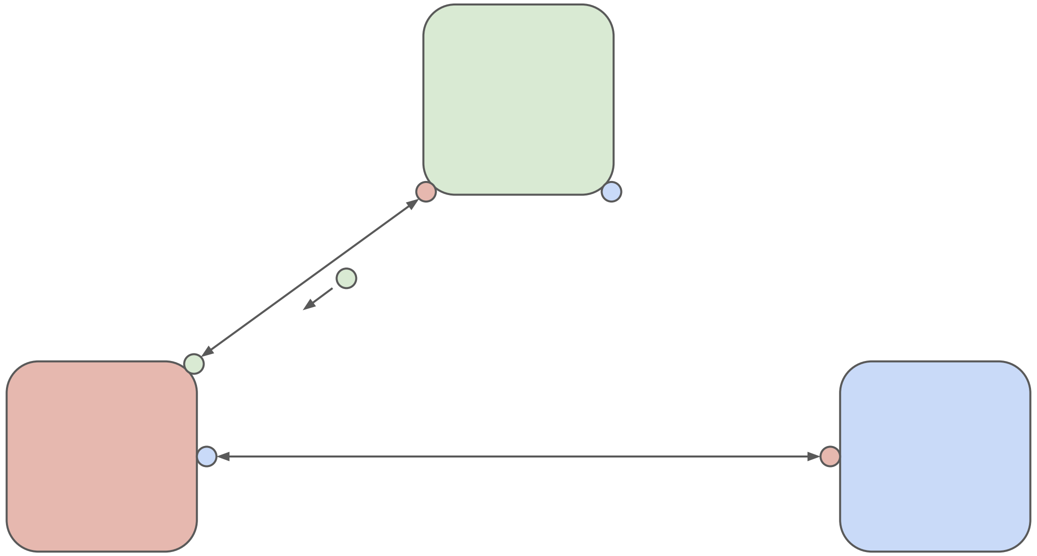
```
void NodeController::OnIntroduce(const ports::NodeName& from_node,
                                 const ports::NodeName& name,
                                 PlatformHandle channel_handle,
                                 const uint64_t remote_capabilities) {
  DCHECK(io_task_runner_->RunsTasksInCurrentSequence());

  if (broker_name_ == ports::kInvalidNodeName || from_node != broker_name_) {
    DVLOG(1) << "Ignoring introduction from non-broker process.";
    DropPeer(from_node, nullptr);
    return;
  }

```

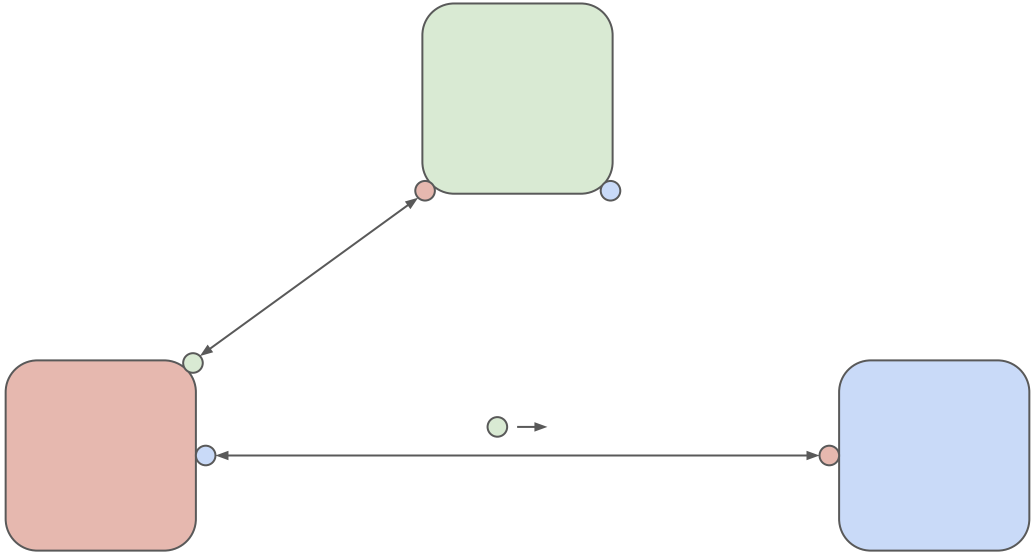


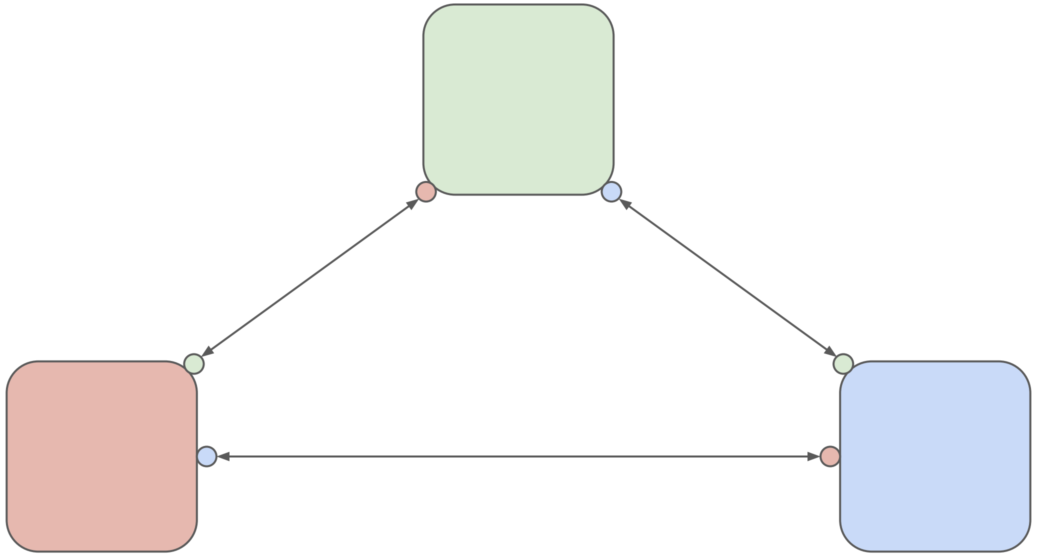


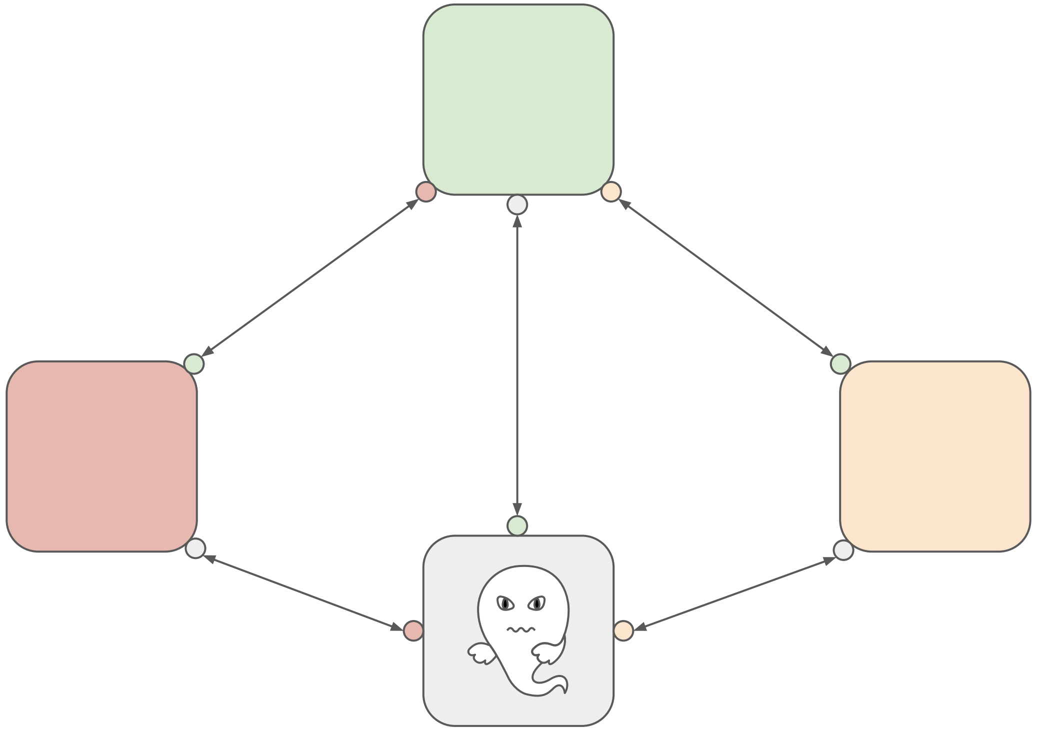


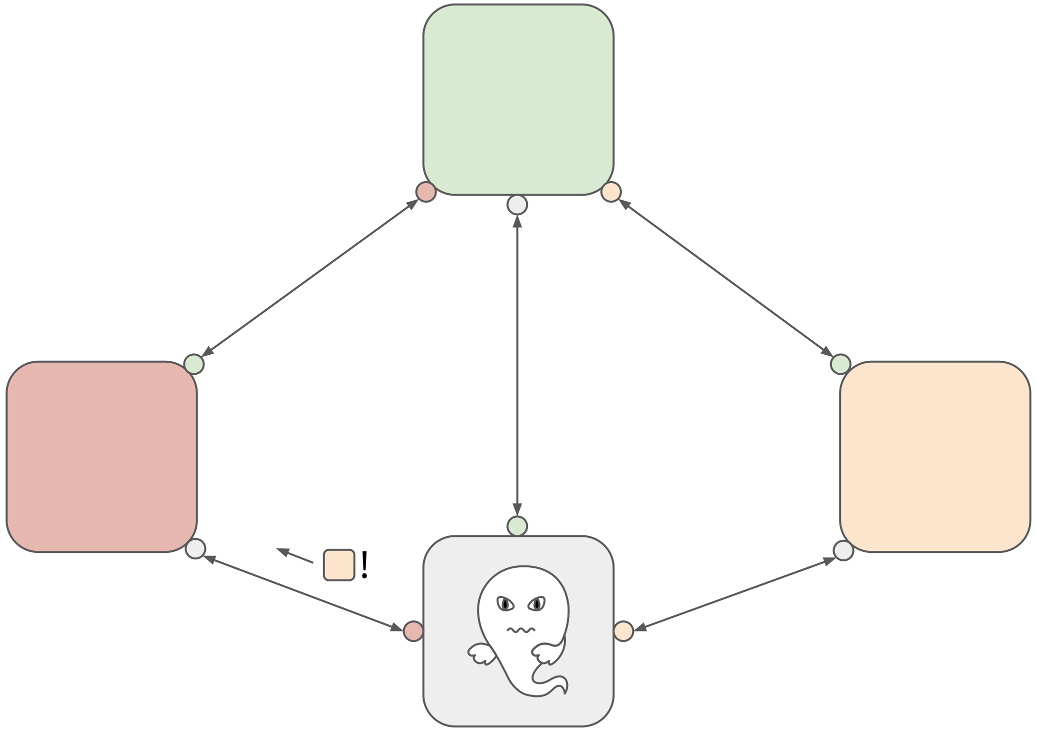


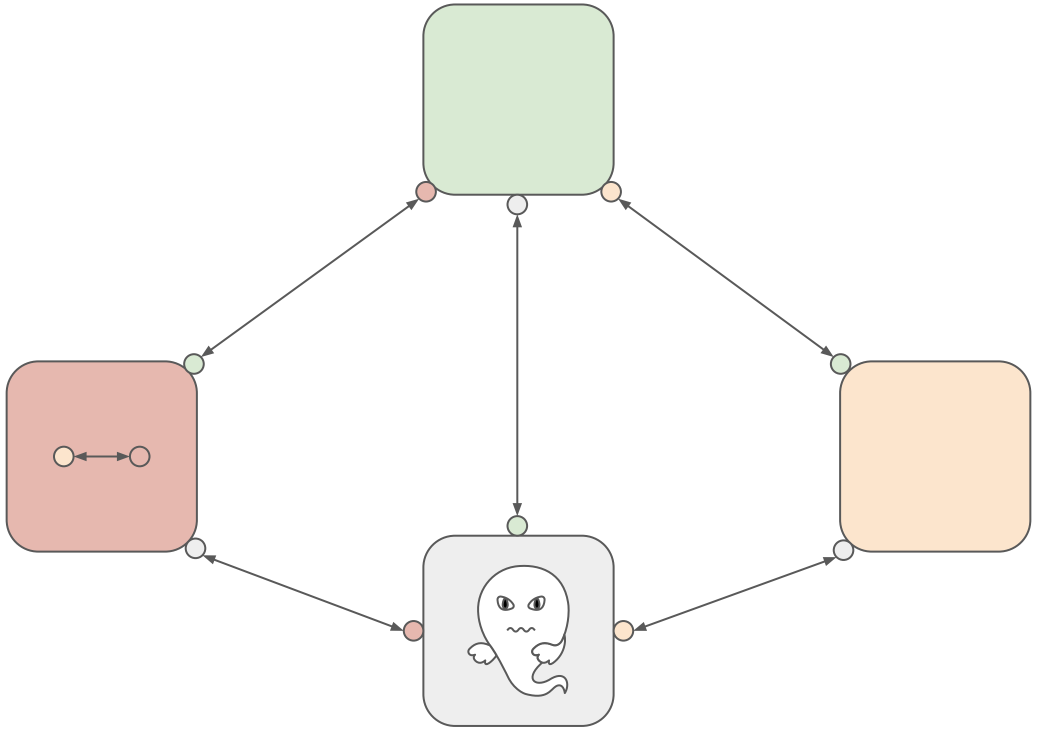


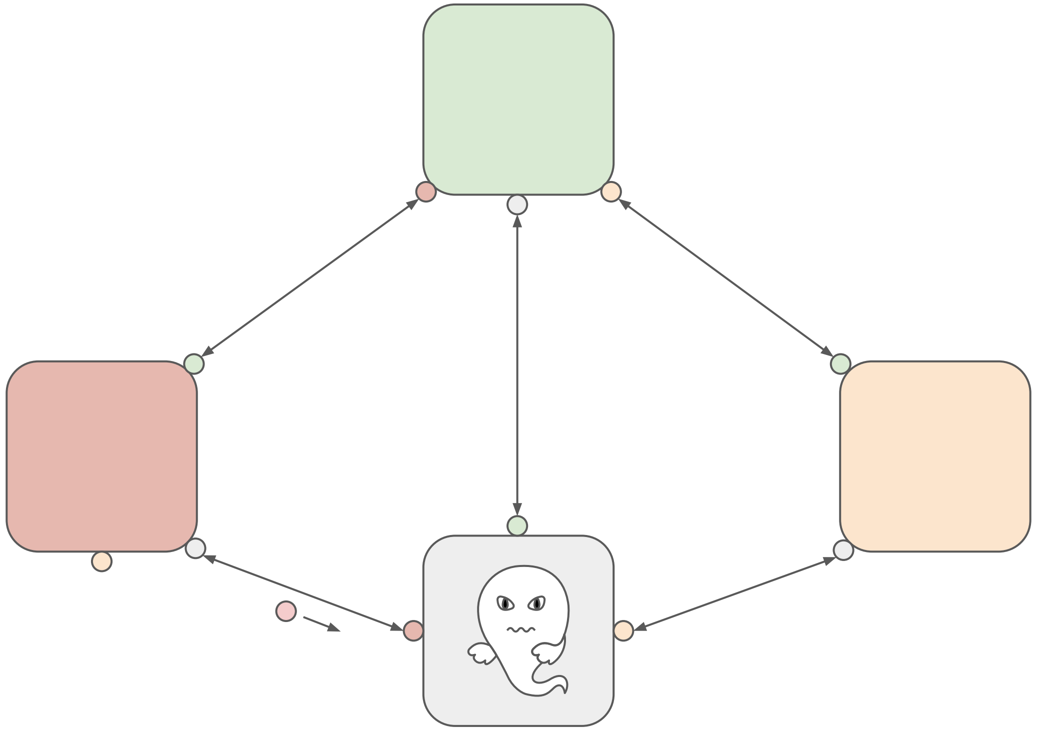


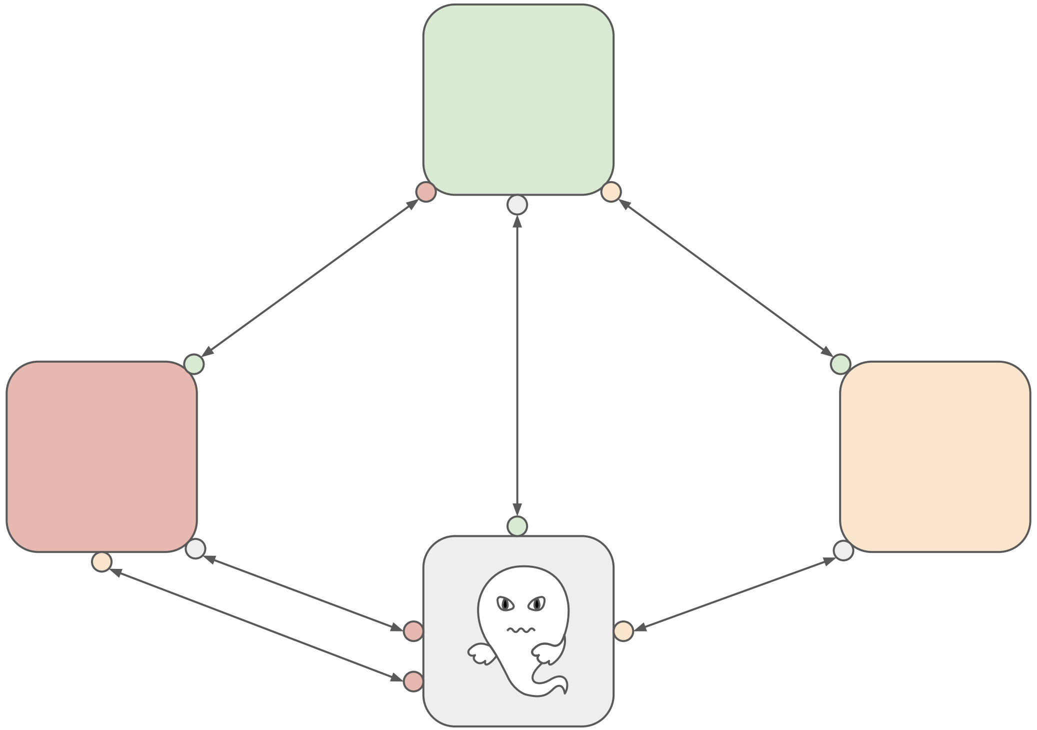


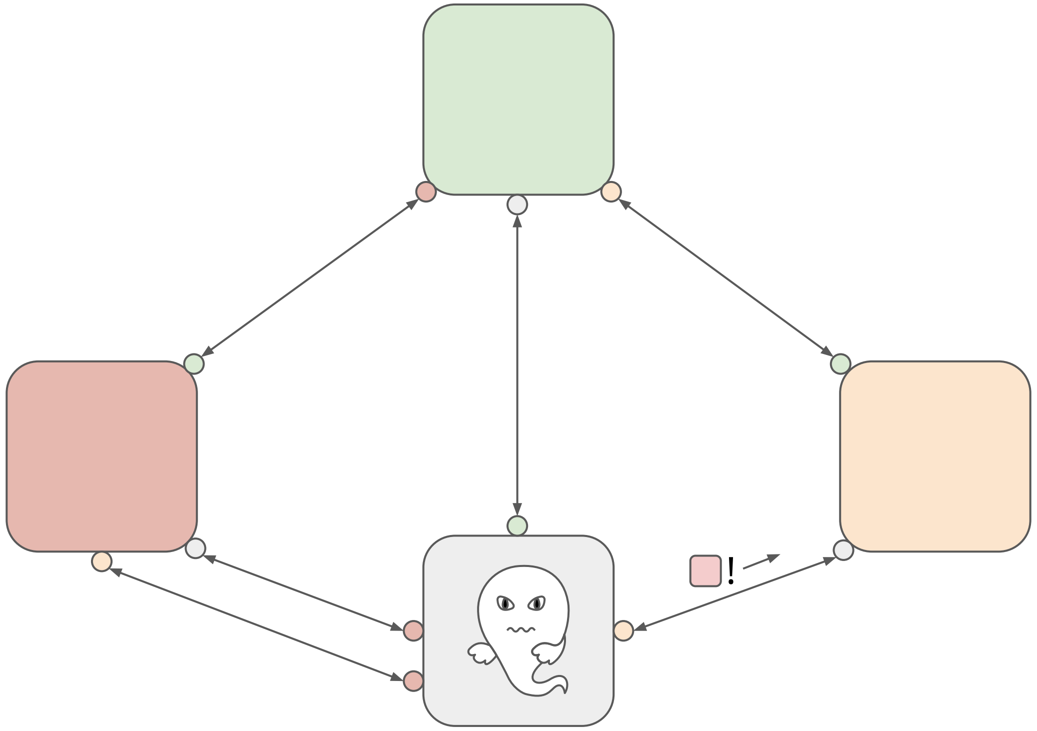


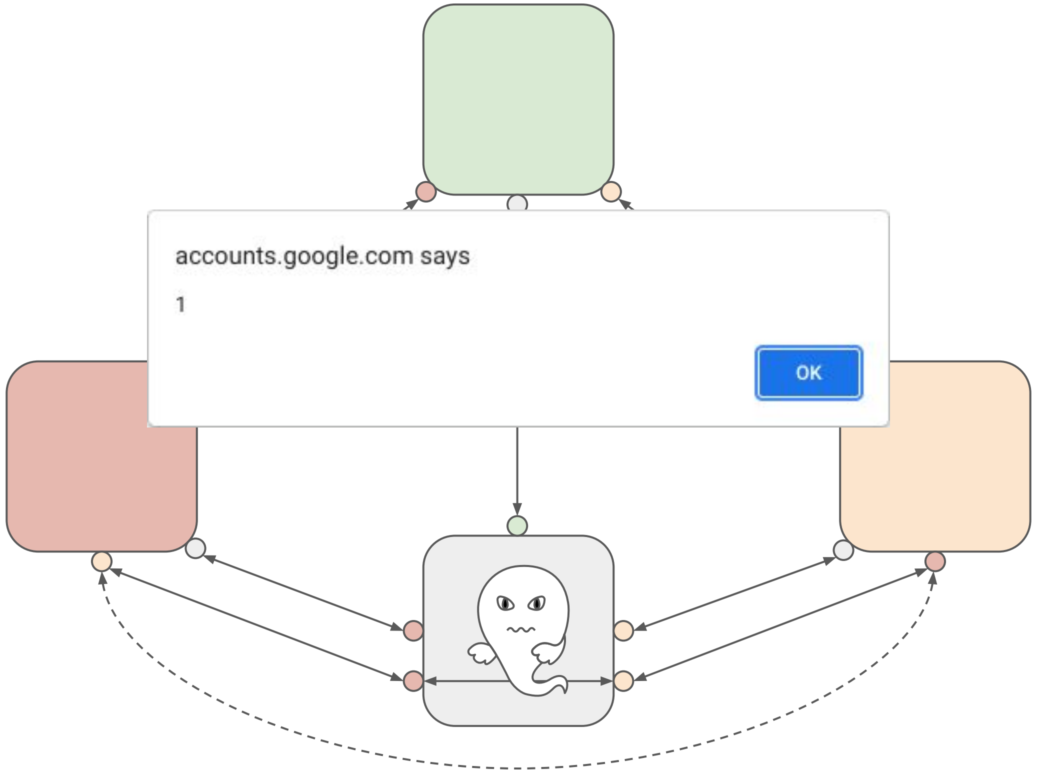






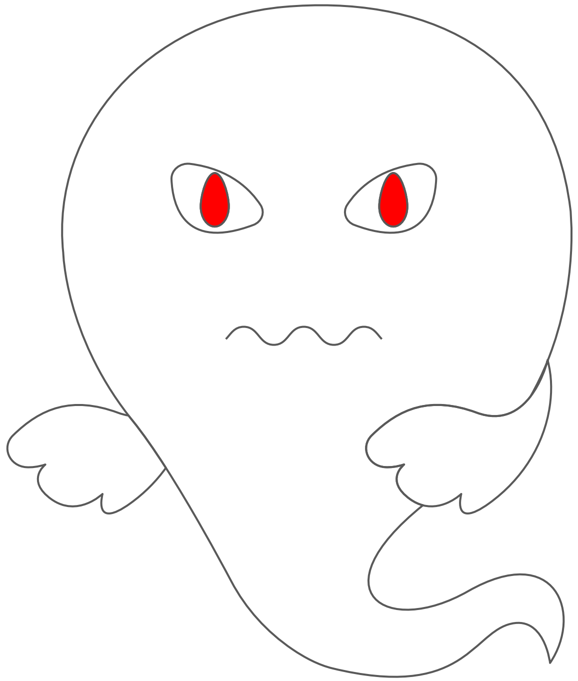


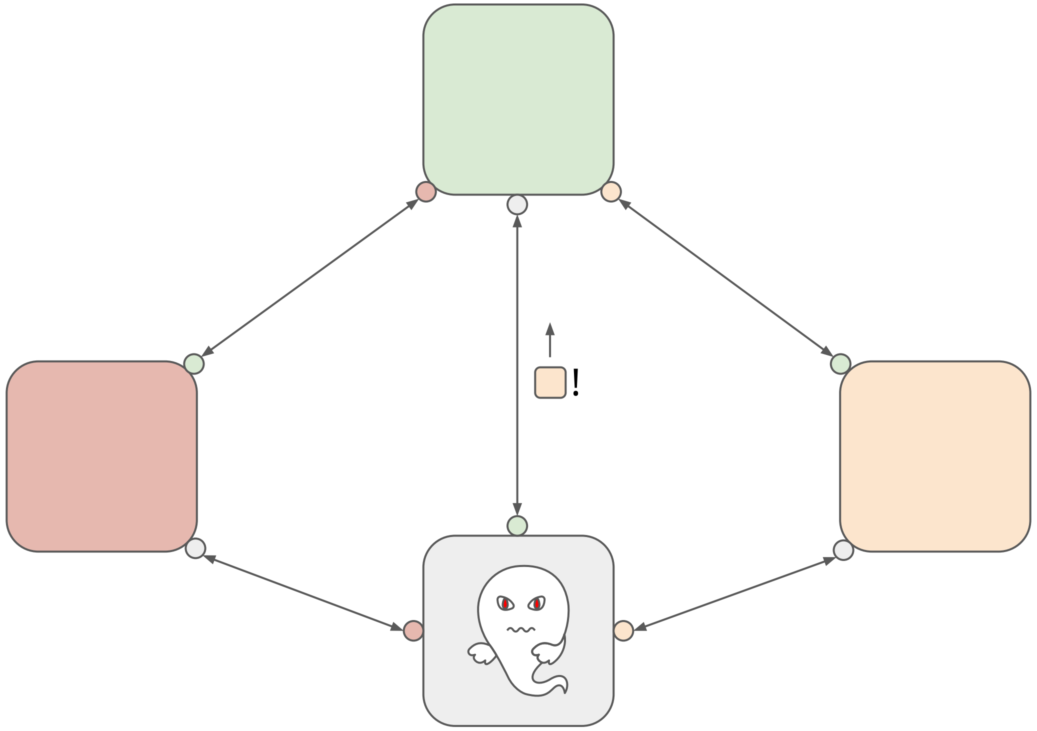


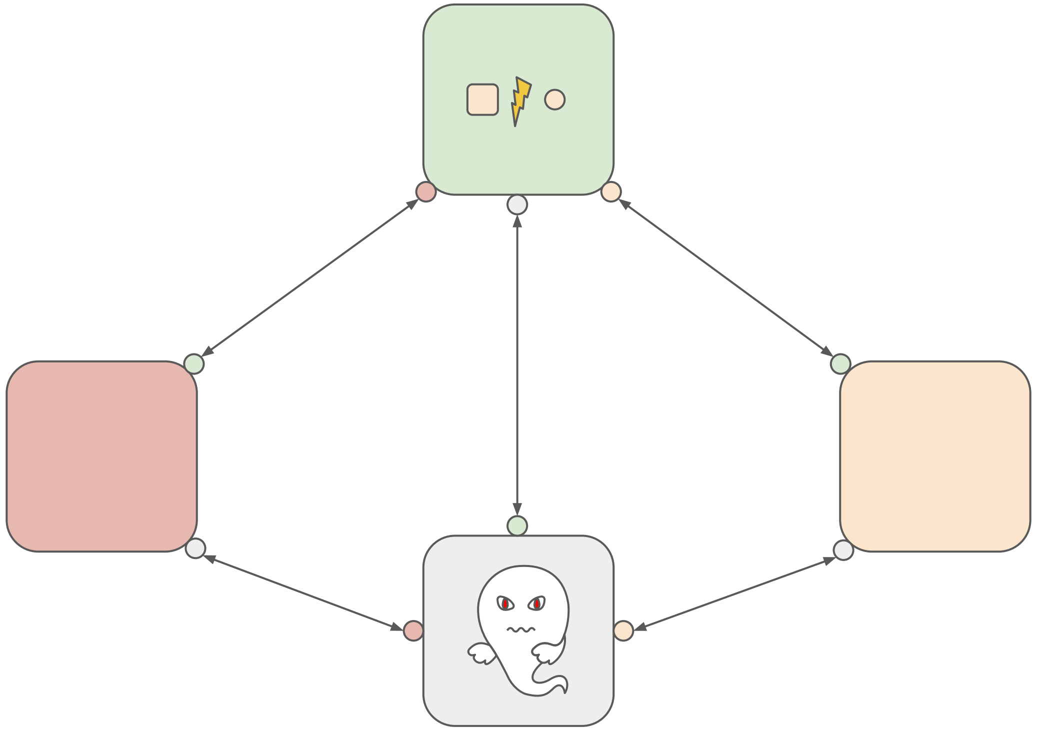


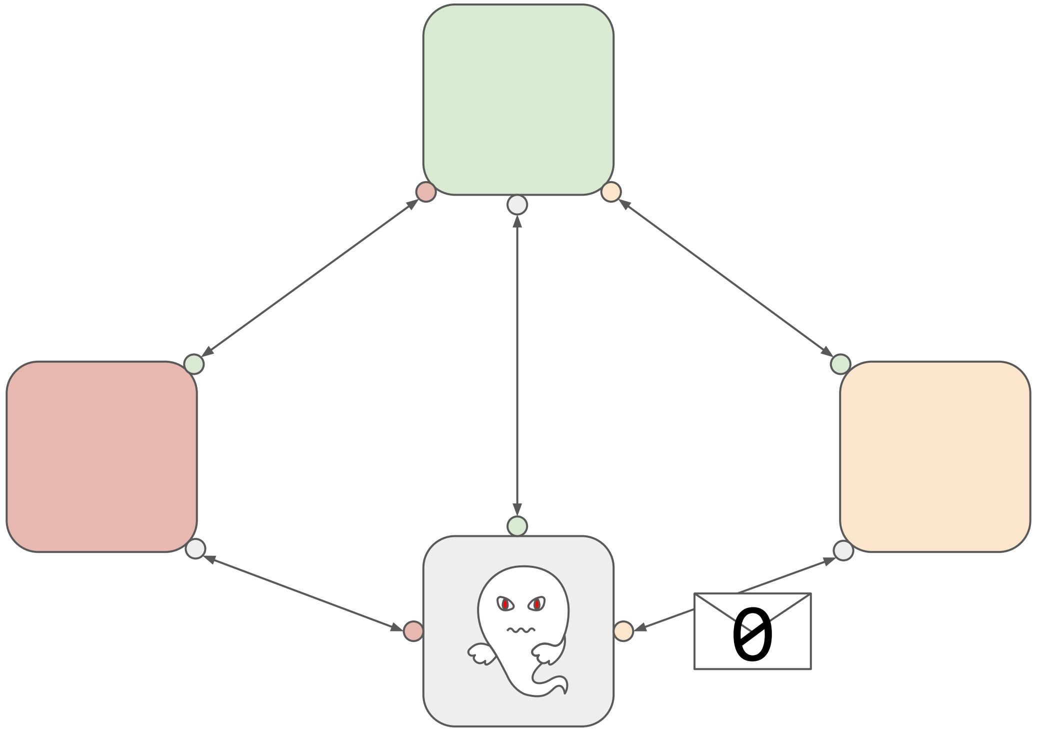
accounts.google.com says

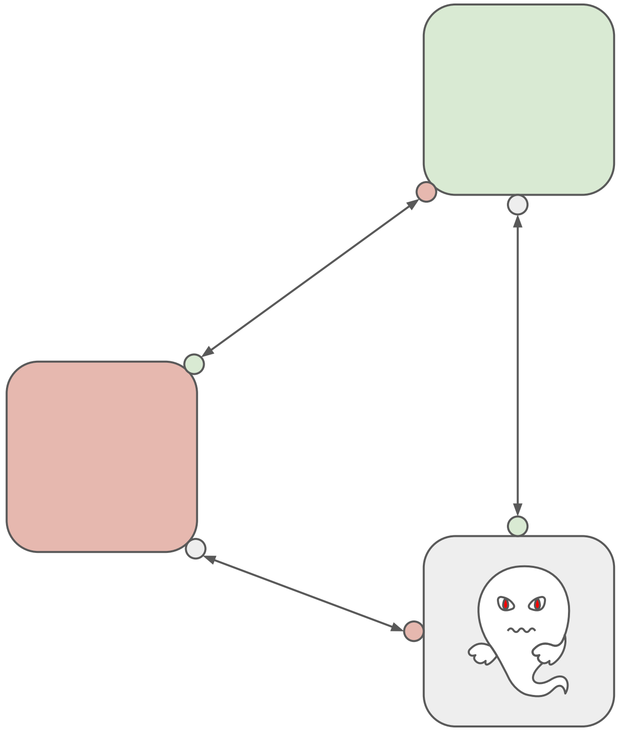
1

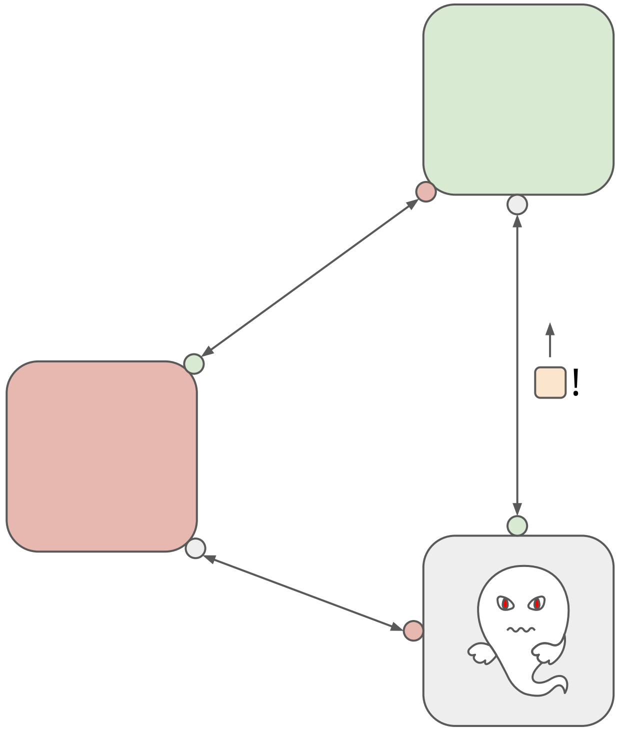


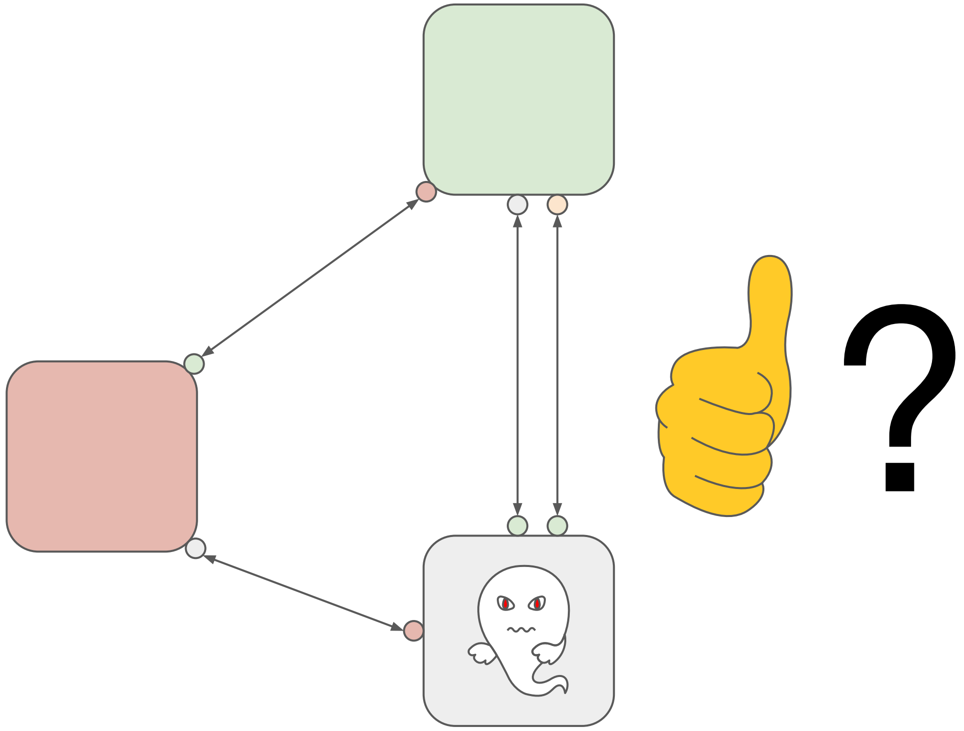


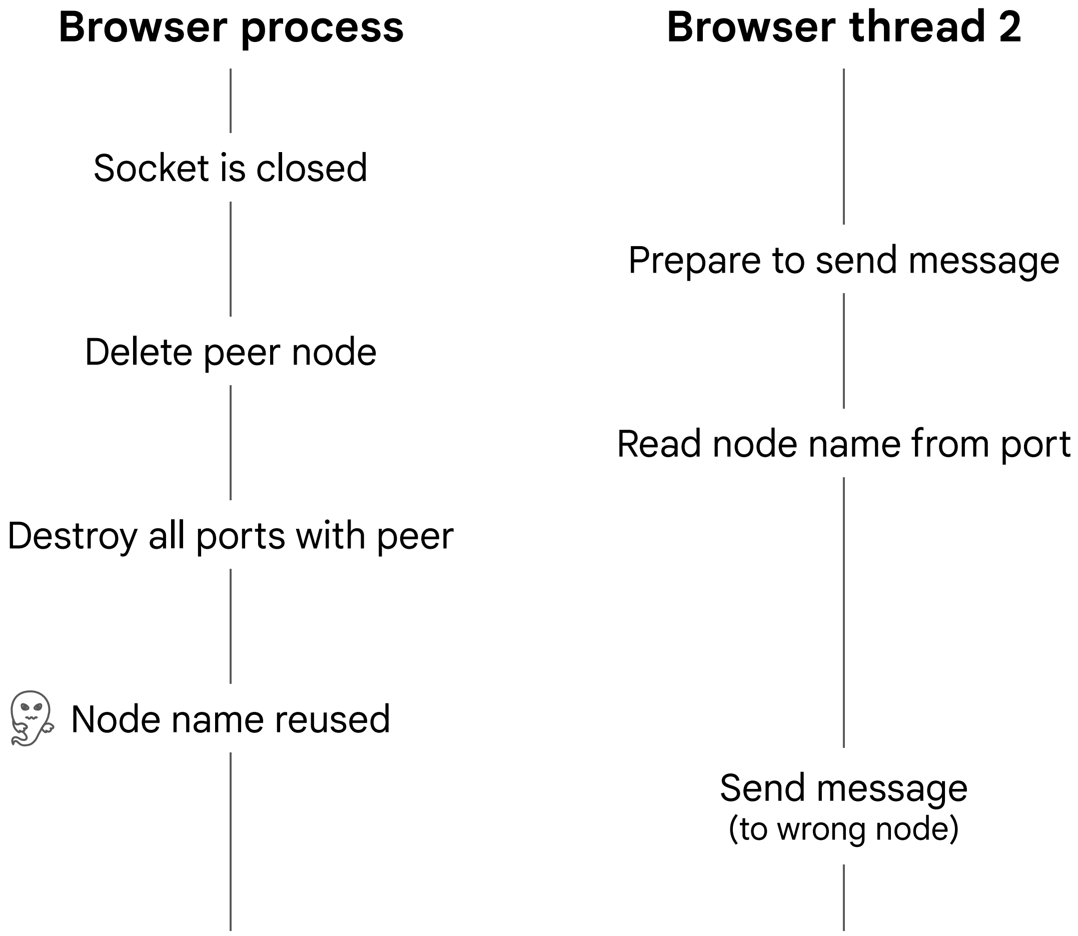


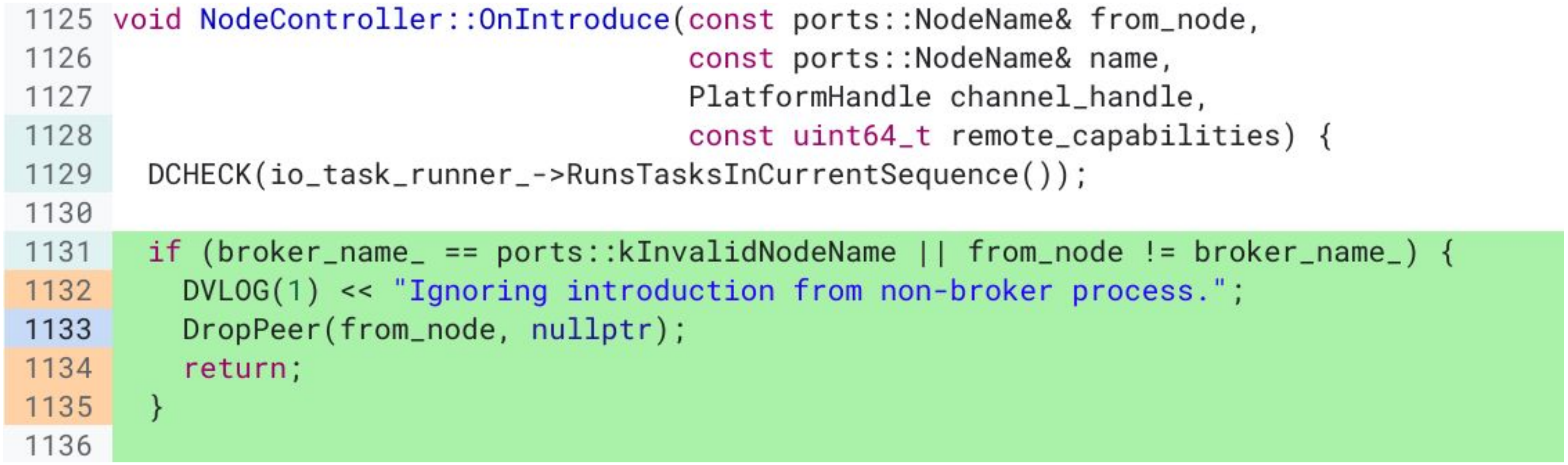

0



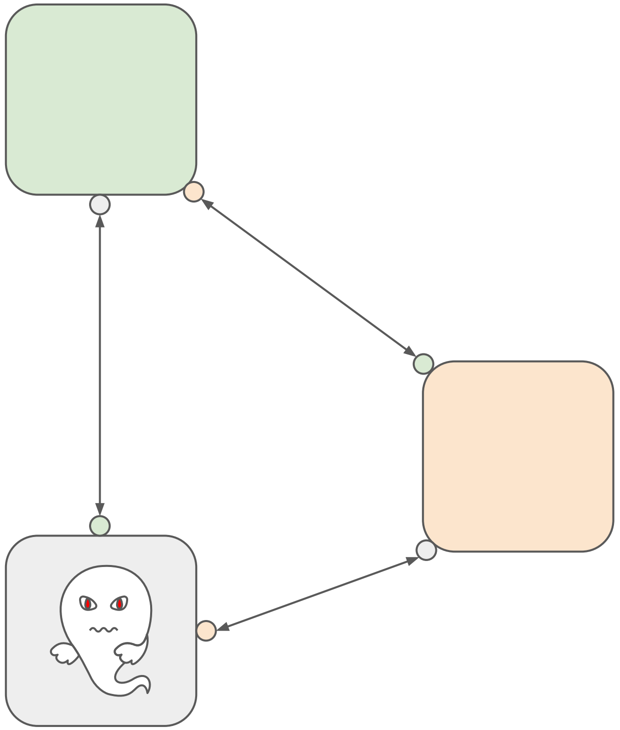


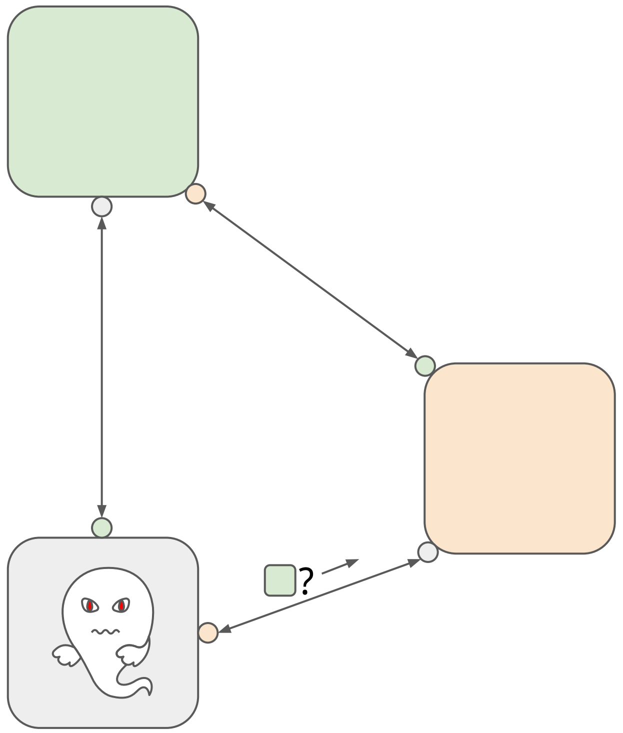


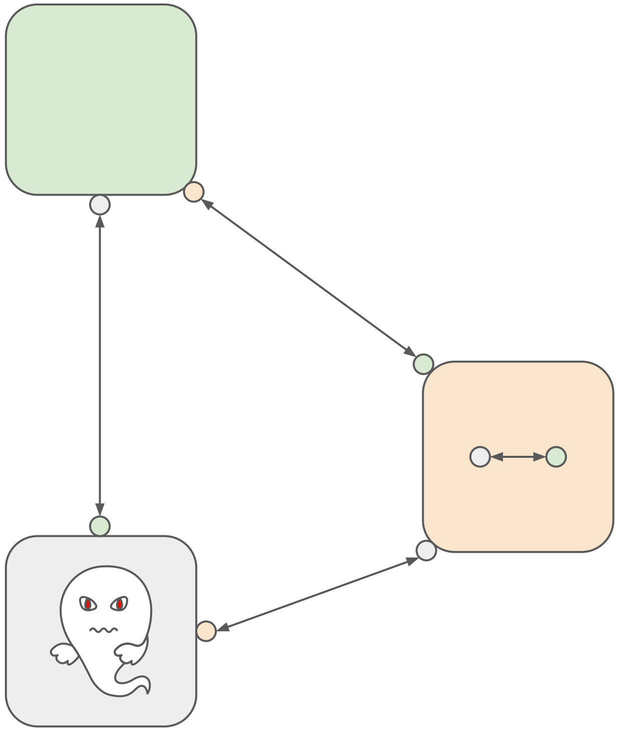

Browser process
Browser thread 2
Socket is closed
Prepare to send message
Delete peer node
Read node name from port
Destroy all ports with peer
Node name reused
Send message
(to wrong node)

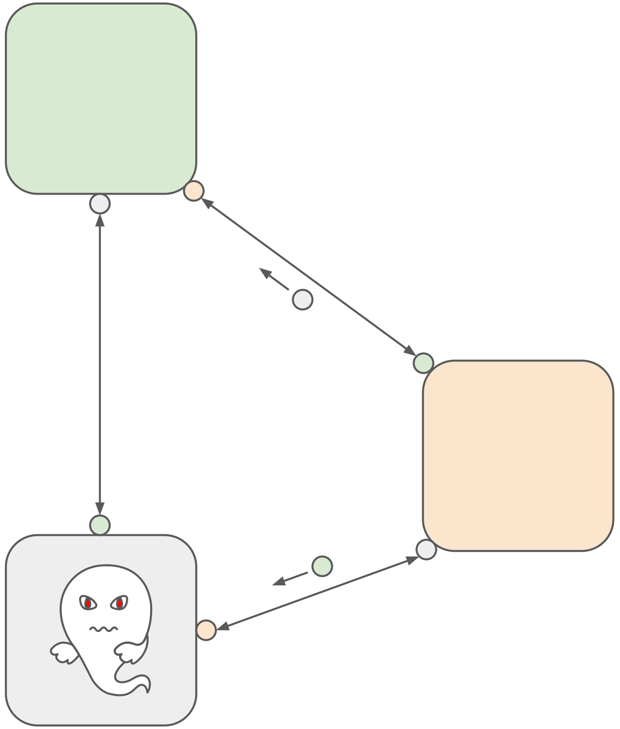


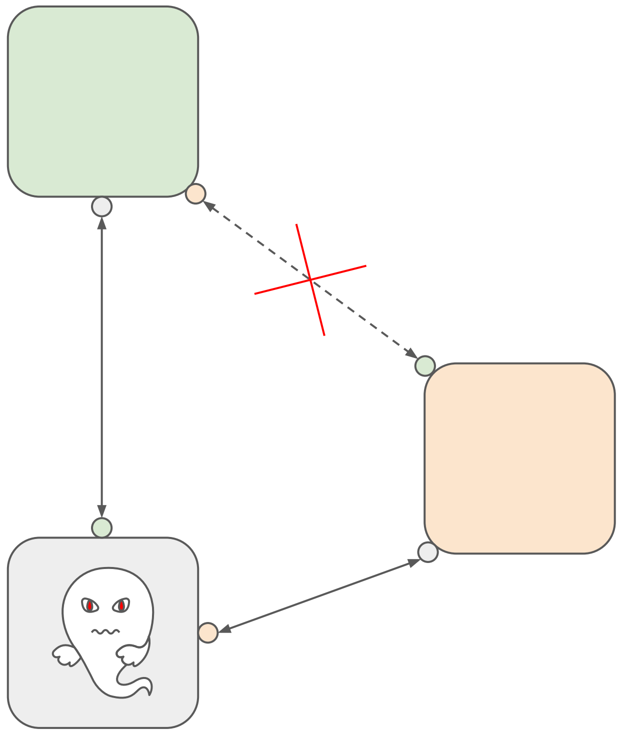
```cpp
void NodeController::OnIntroduce(const ports::NodeName& from_node,
                                 const ports::NodeName& name,
                                 PlatformHandle channel_handle,
                                 const uint64_t remote_capabilities) {
  DCHECK(io_task_runner_->RunsTasksInCurrentSequence());

  if (broker_name_ == ports::kInvalidNodeName || from_node != broker_name_) {
    DVLOG(1) << "Ignoring introduction from non-broker process.";
    DropPeer(from_node, nullptr);
    return;
  }

```

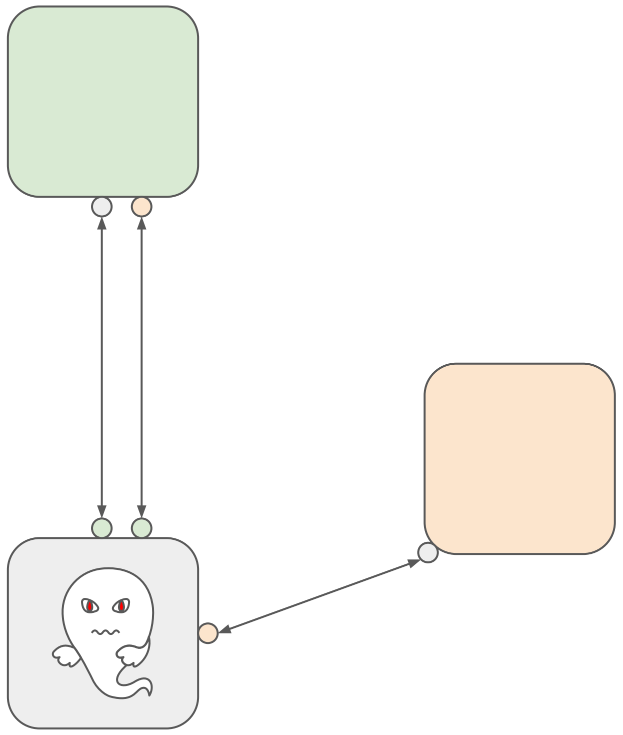


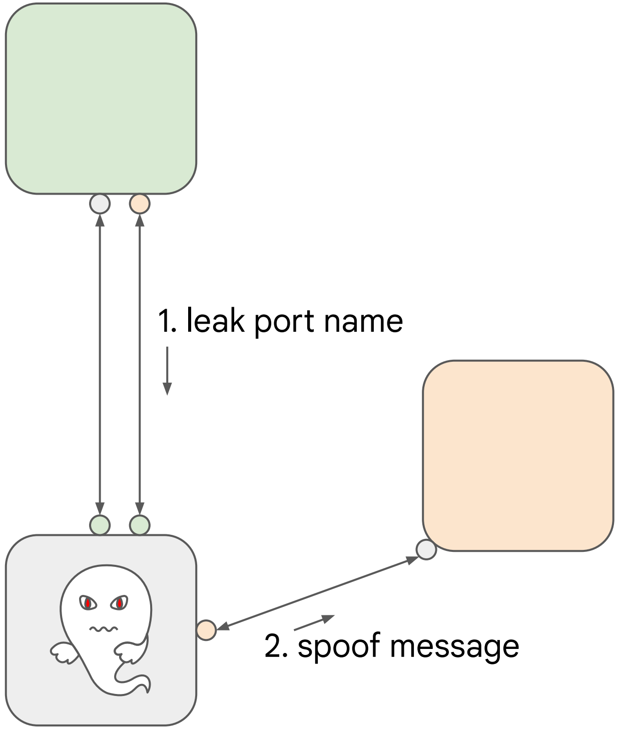











1. leak port name
2. spoof message

Browser process

Socket is closed

Delete peer node

Destroy all ports with peer

Port OnError handlers
Send PortClosed events
ChildProcessHost: `kill(child)`

👻 Node name reused

Network Process

Socket is closed

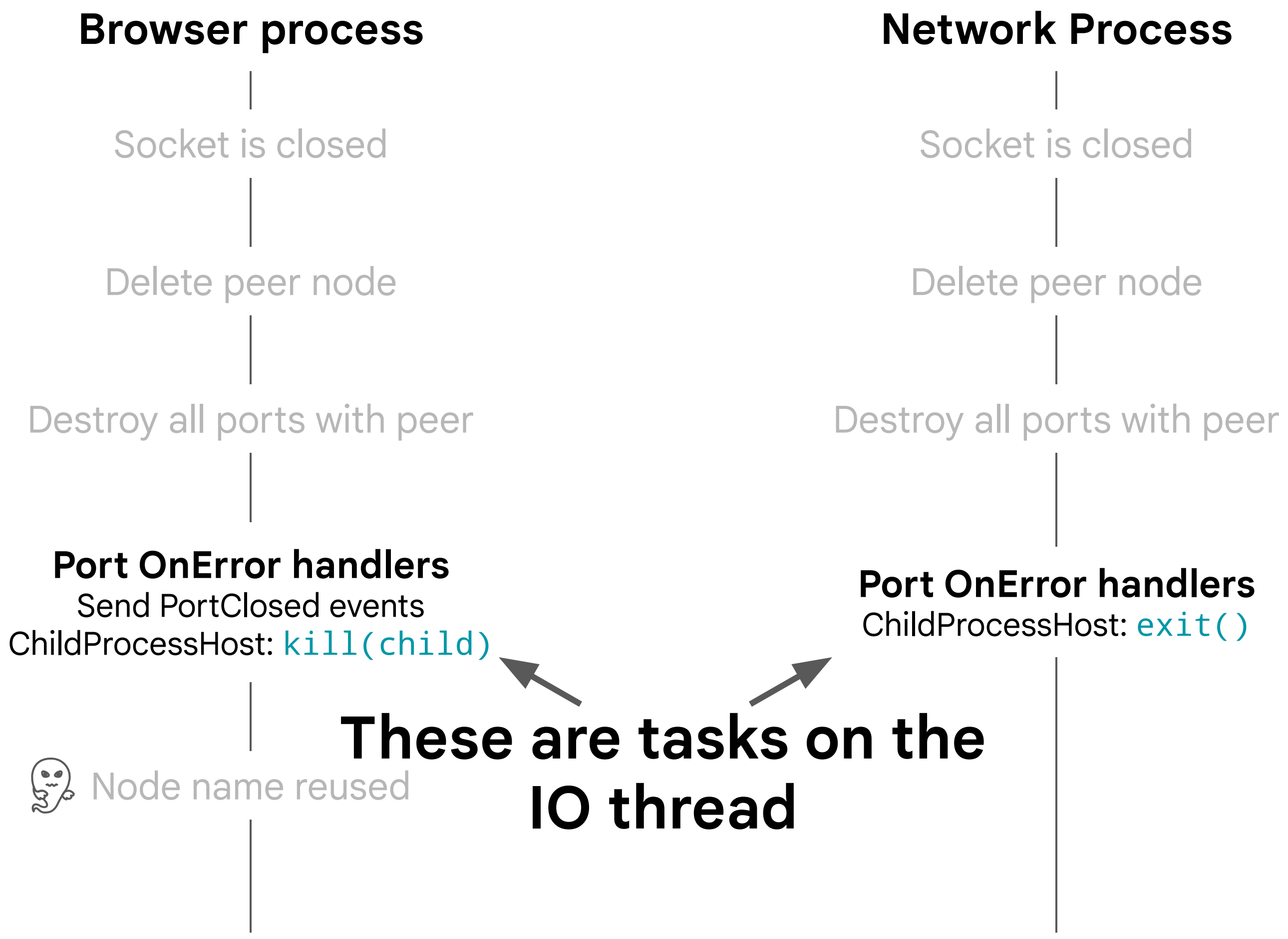
Delete peer node

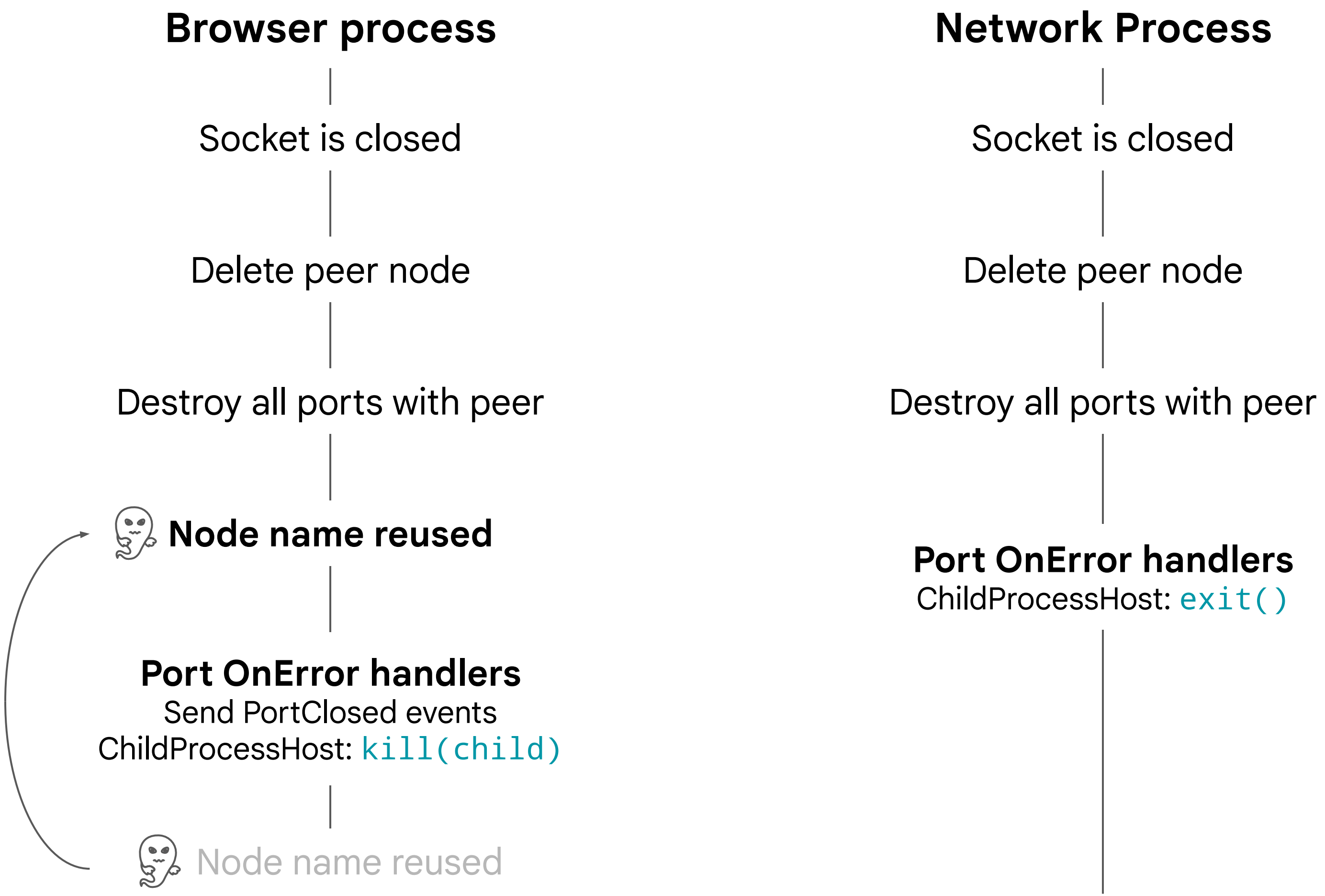
Destroy all ports with peer

Port OnError handlers
ChildProcessHost: `exit()`

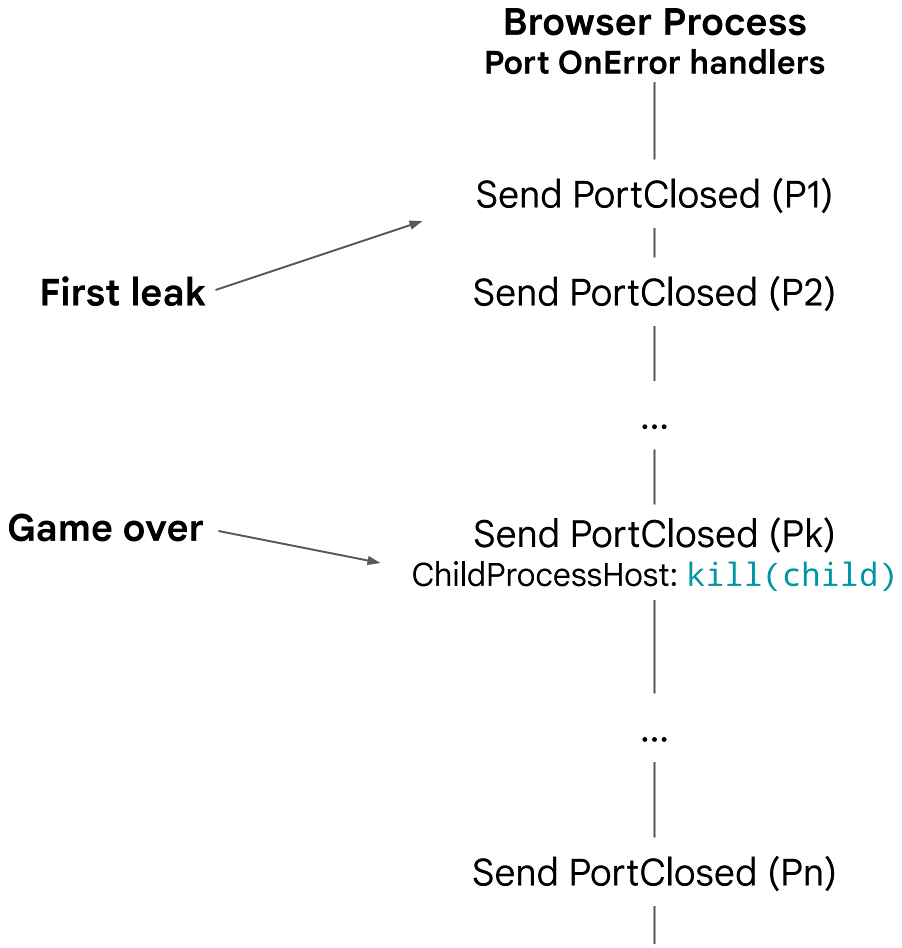
"I have a dangerous fascination with terrible bugs."

@halvarflake


Browser process
Socket is closed
Delete peer node
Destroy all ports with peer
Port OnError handlers
Send PortClosed events
ChildProcessHost: kill(child)
Node name reused
Network Process
Socket is closed
Delete peer node
Destroy all ports with peer
Port OnError handlers
ChildProcessHost: exit()
These are tasks on the IO thread



Browser process
Socket is closed
Delete peer node
Destroy all ports with peer
Node name reused
Port OnError handlers
Send PortClosed events
ChildProcessHost: `kill(child)`
Node name reused
Network Process
Socket is closed
Delete peer node
Destroy all ports with peer
Port OnError handlers
ChildProcessHost: `exit()`


Browser Process
Port OnError handlers
Send PortClosed (P1)
Send PortClosed (P2)
...
Send PortClosed (Pk)
ChildProcessHost: kill(child)
...
Send PortClosed (Pn)
First leak
Game over

Issues:

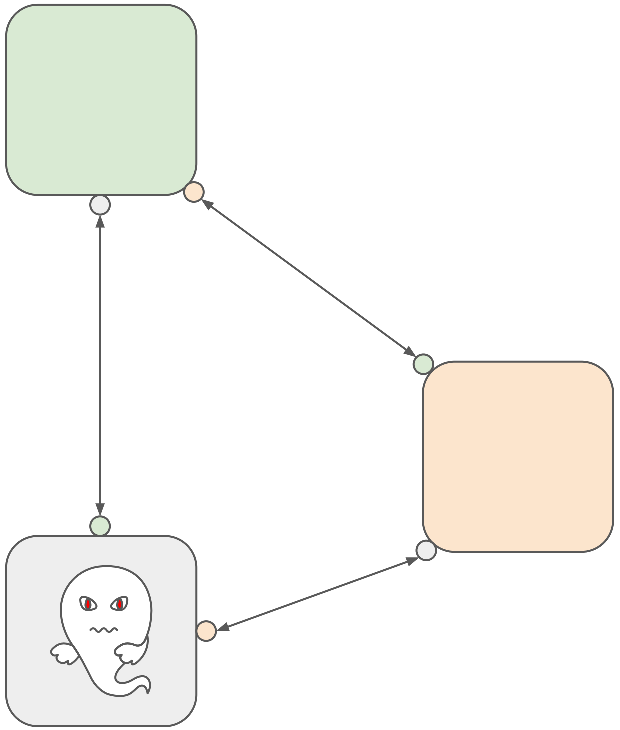
- Tight race between leak and kill
- Network process will *exit()*

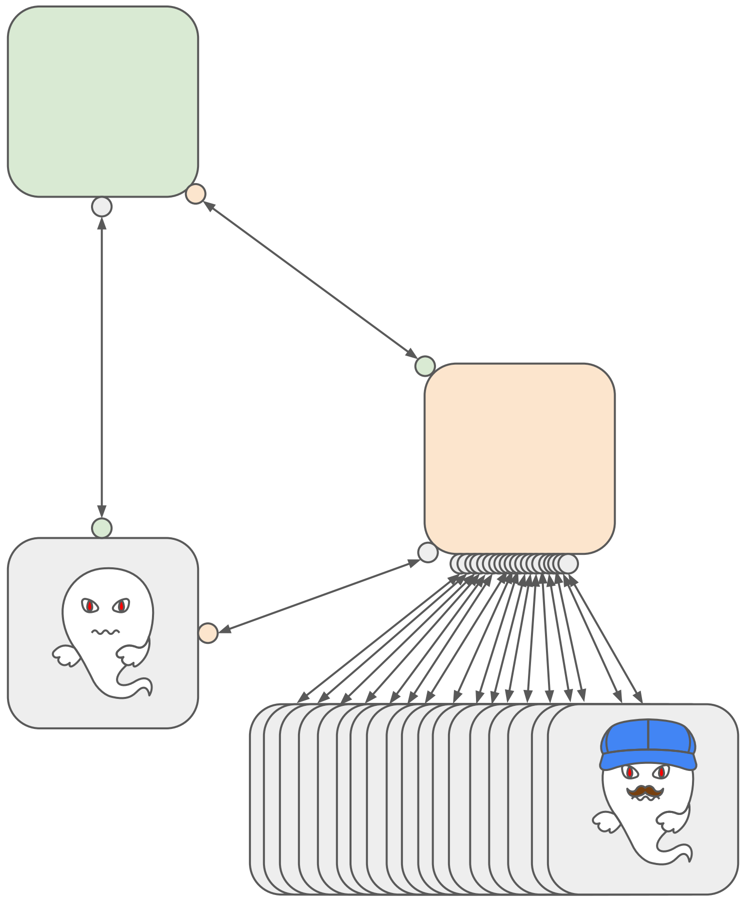
KATE
We need your help to overload the Gibson.

RAZOR
You are going to need more than just two media icons like us. You need an army.

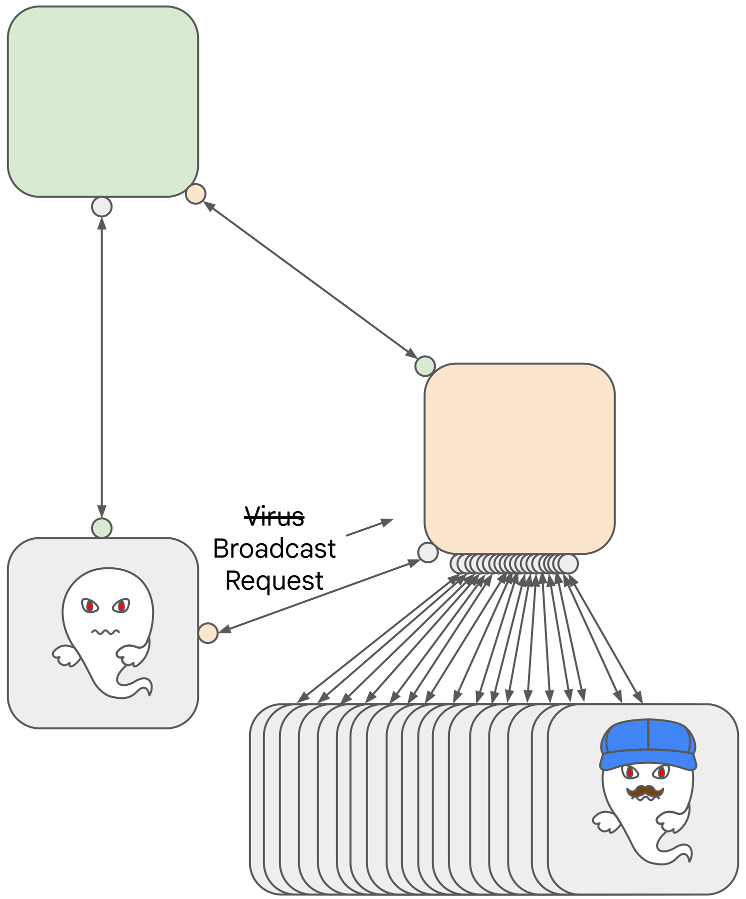
BLADE
That's it! An electronic army! If I were us, I'd get on the internet, send out a major distress signal.

RAZOR
Hackers of the World, Unite!





KATE
Now listen up, use your best viruses to buy us time, we have to spoof a message to the network process.

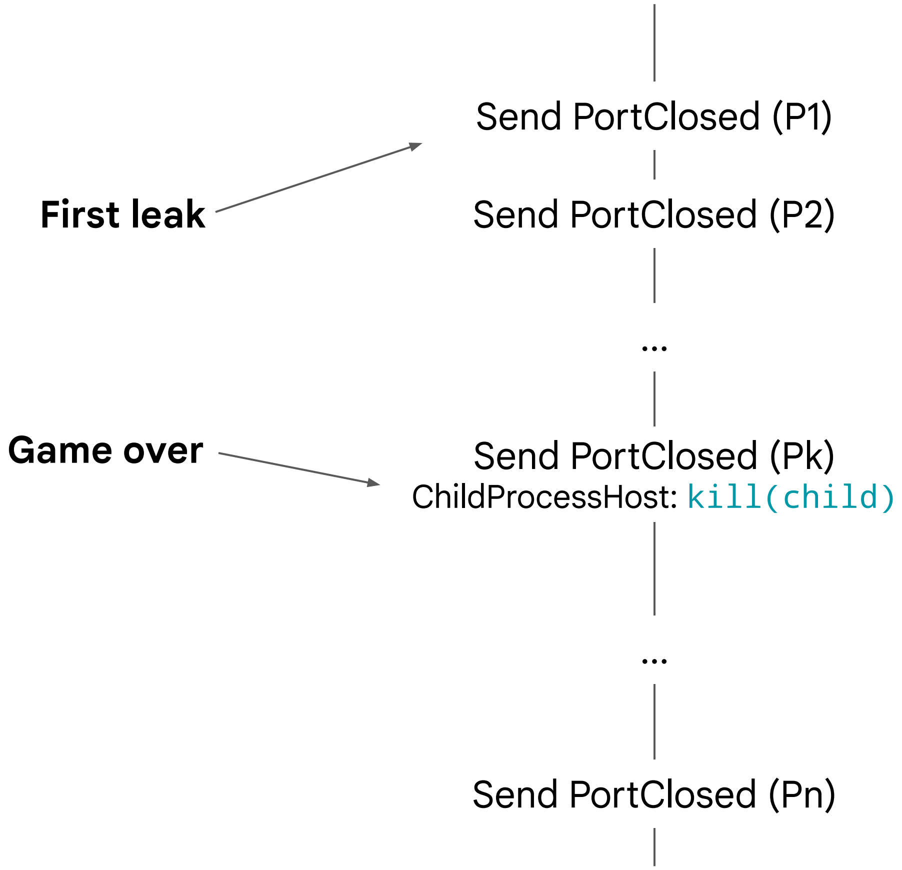

~~Virus~~
Broadcast
Request

Issues:

- Tight race between leak and kill
- ~~Network process will *exit()*~~

KATE
It's the Gibson, it's finding us too fast.

Browser Process
Port OnError handlers



Browser Process
Port OnError handlers

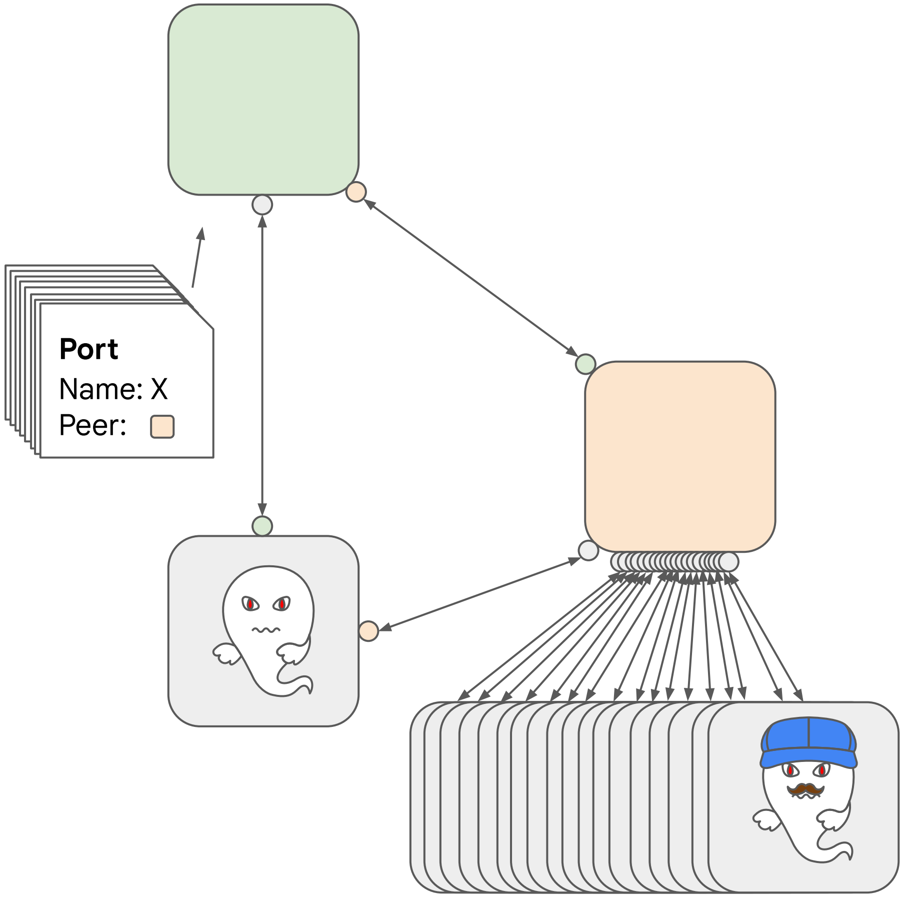
Send PortClosed (P1)

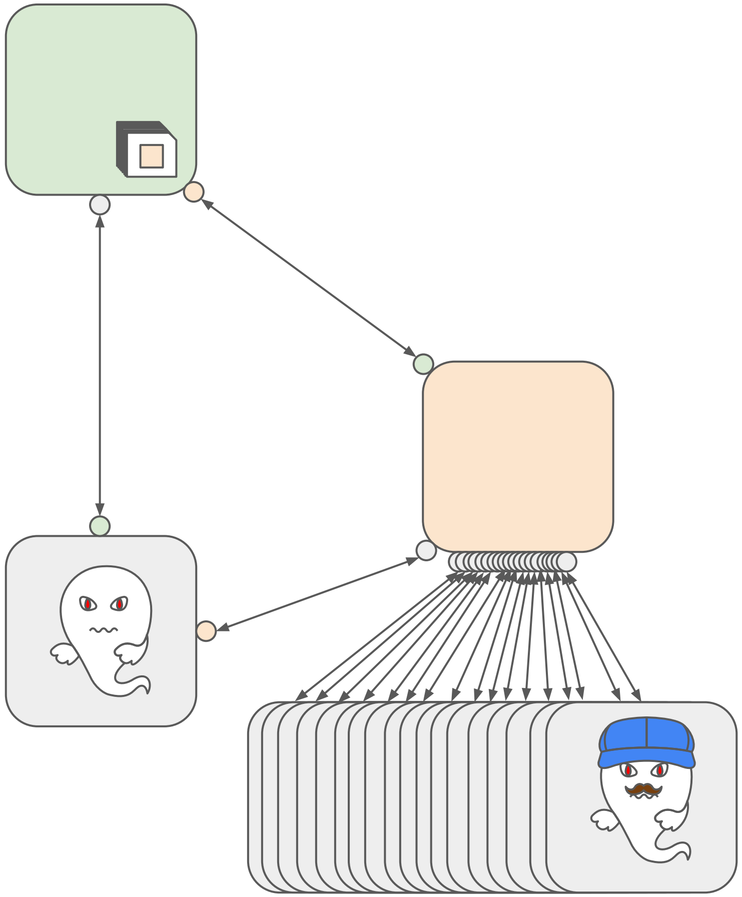
Send PortClosed (P2)

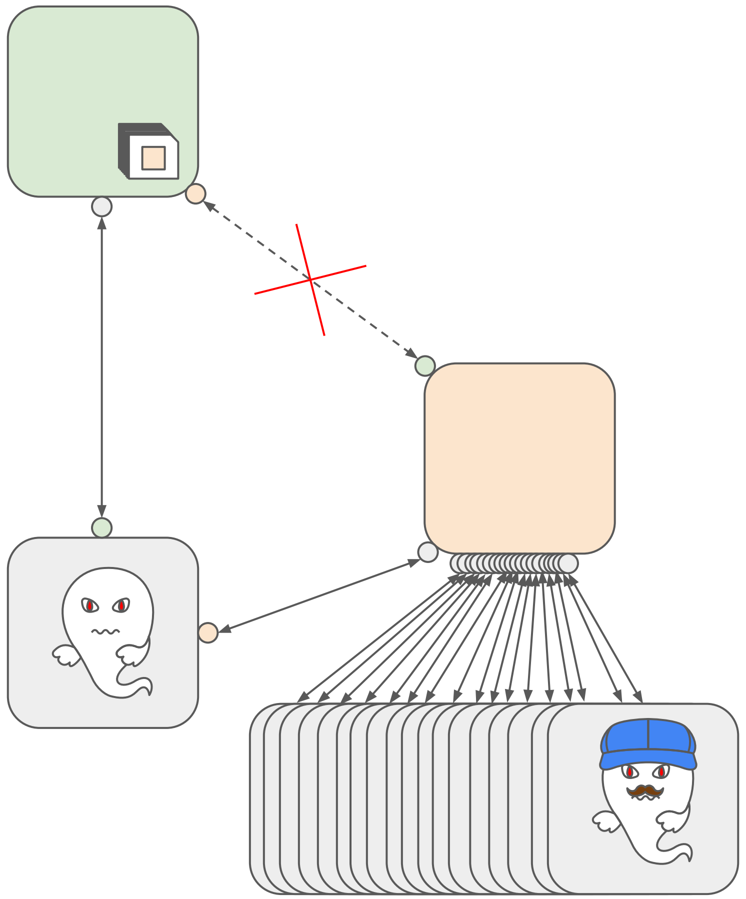
...

Send PortClosed (P **100000**)
ChildProcessHost: `kill(child)`

...


Port
Name: X
Peer:





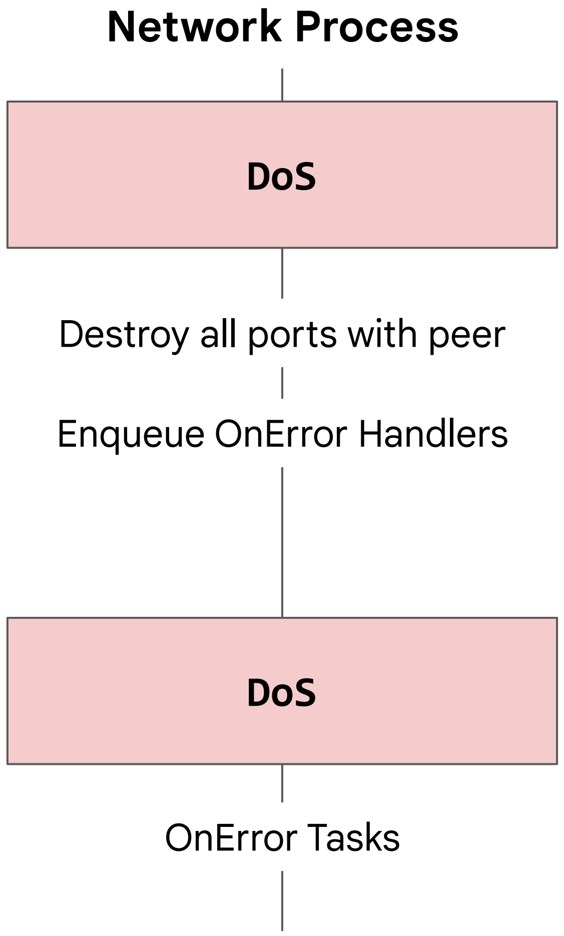
Issues:

- ~~Tight race between leak and kill~~
- ~~Network process will *exit()*~~

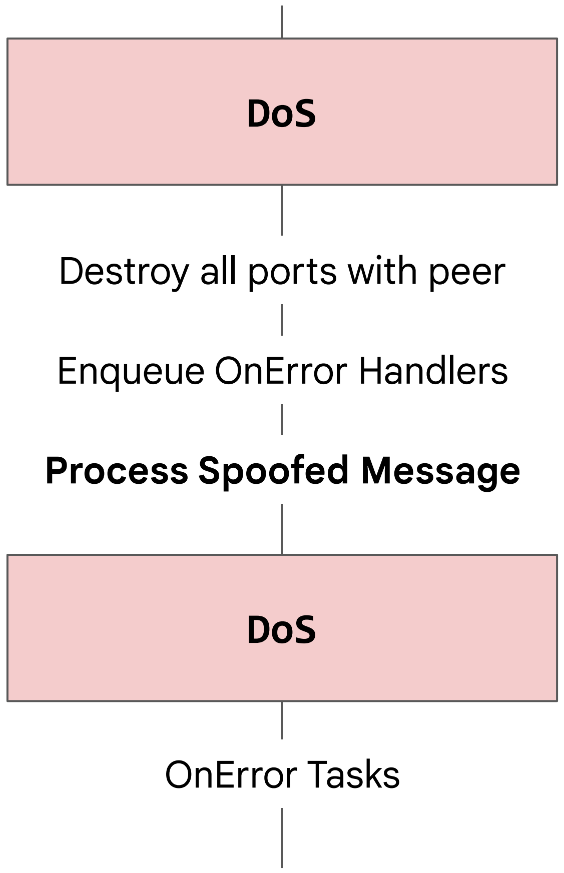
Issues:

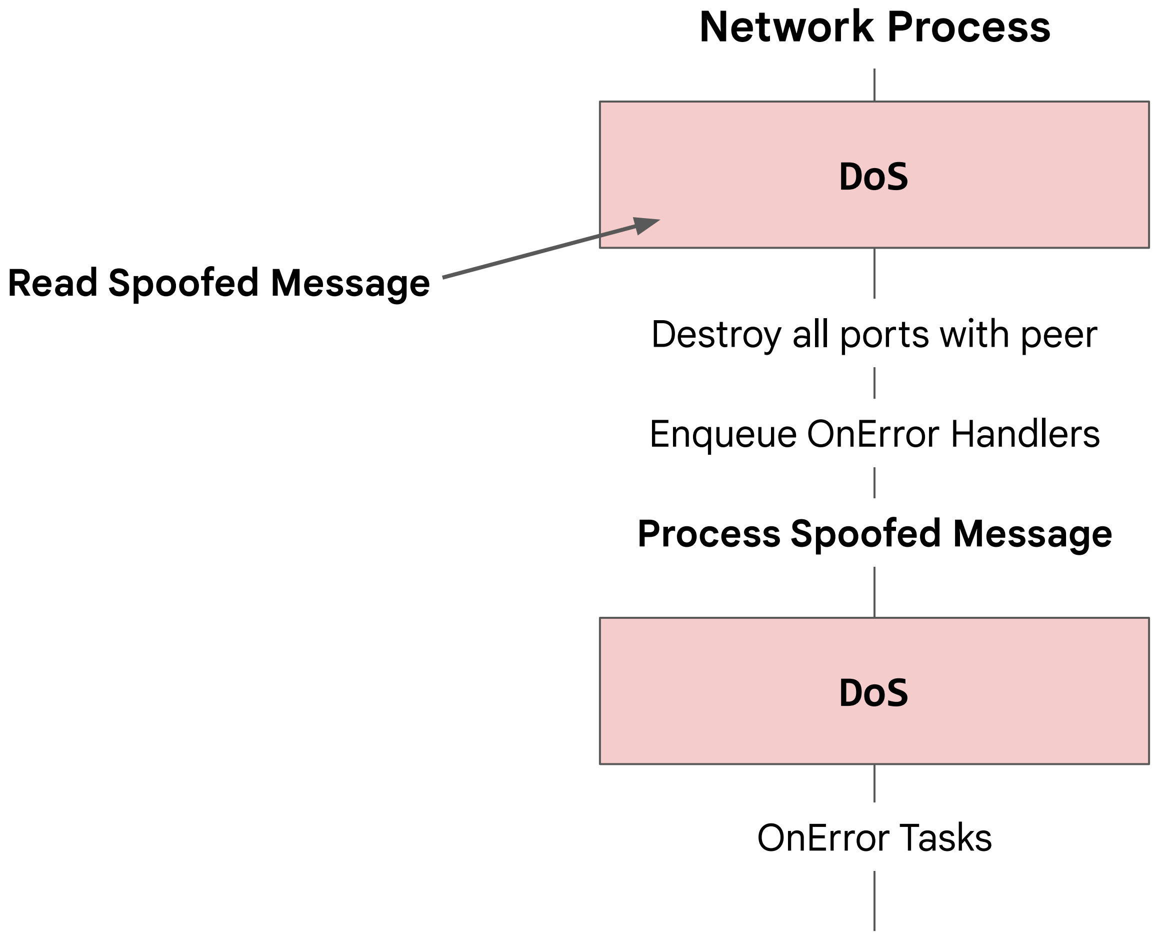
- ~~Tight race between leak and kill~~
- ~~Network process will *exit()*~~
- **How to inject a message during the DoS?**

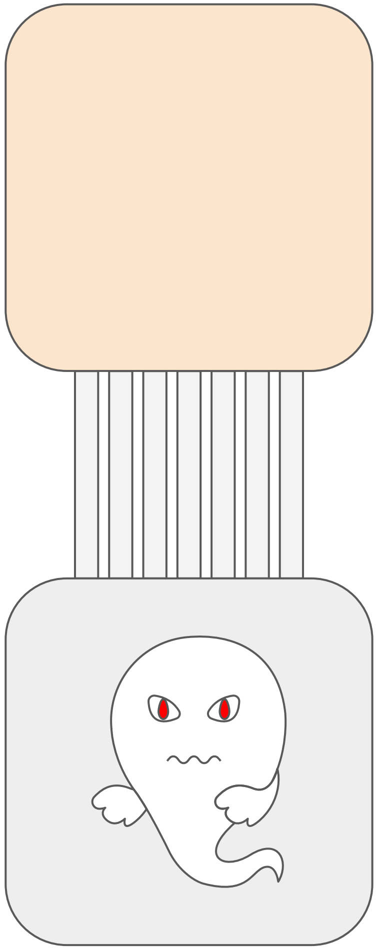
Turn off and on again?

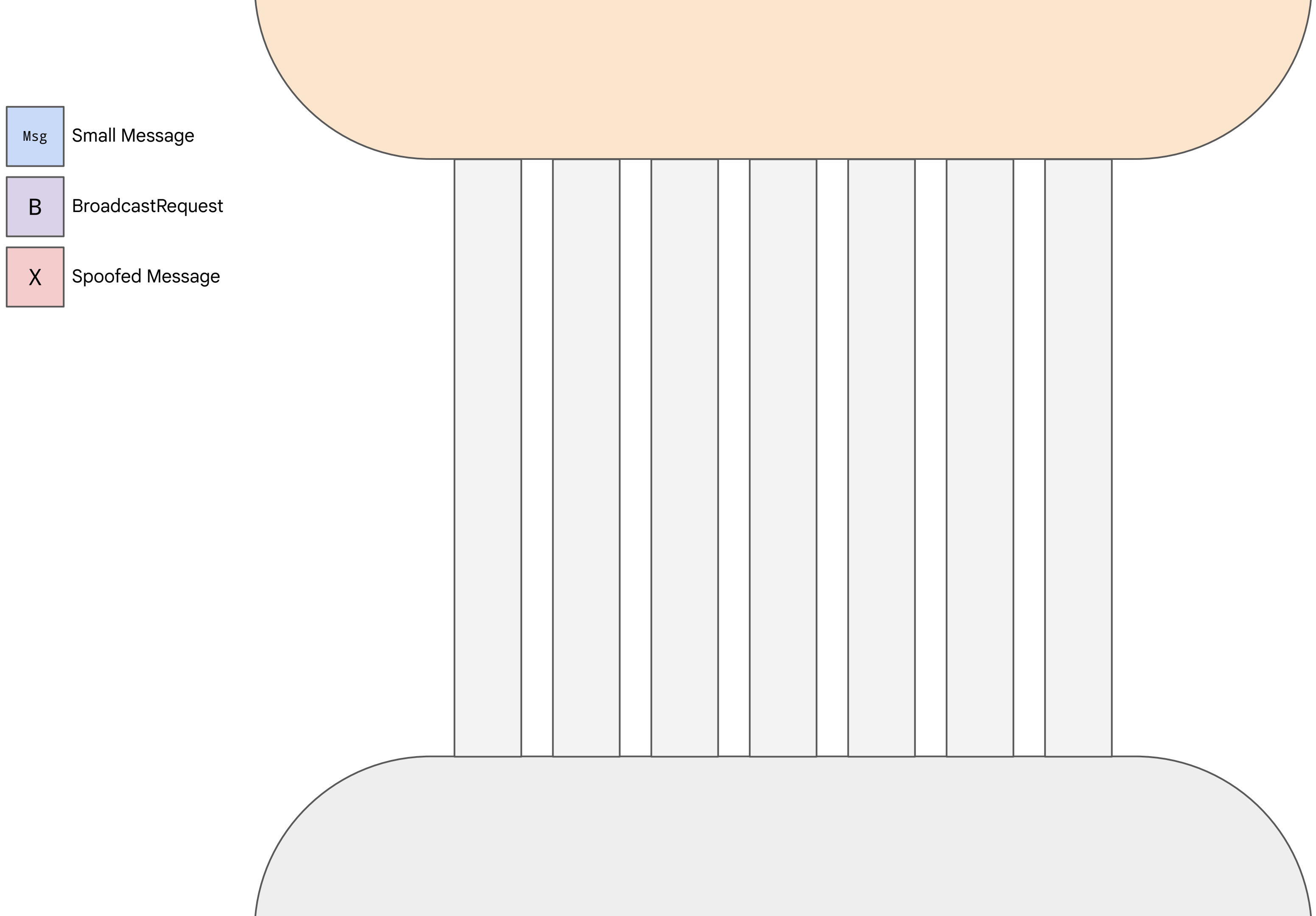

Network Process
DoS
Destroy all ports with peer
Enqueue OnError Handlers
DoS
OnError Tasks

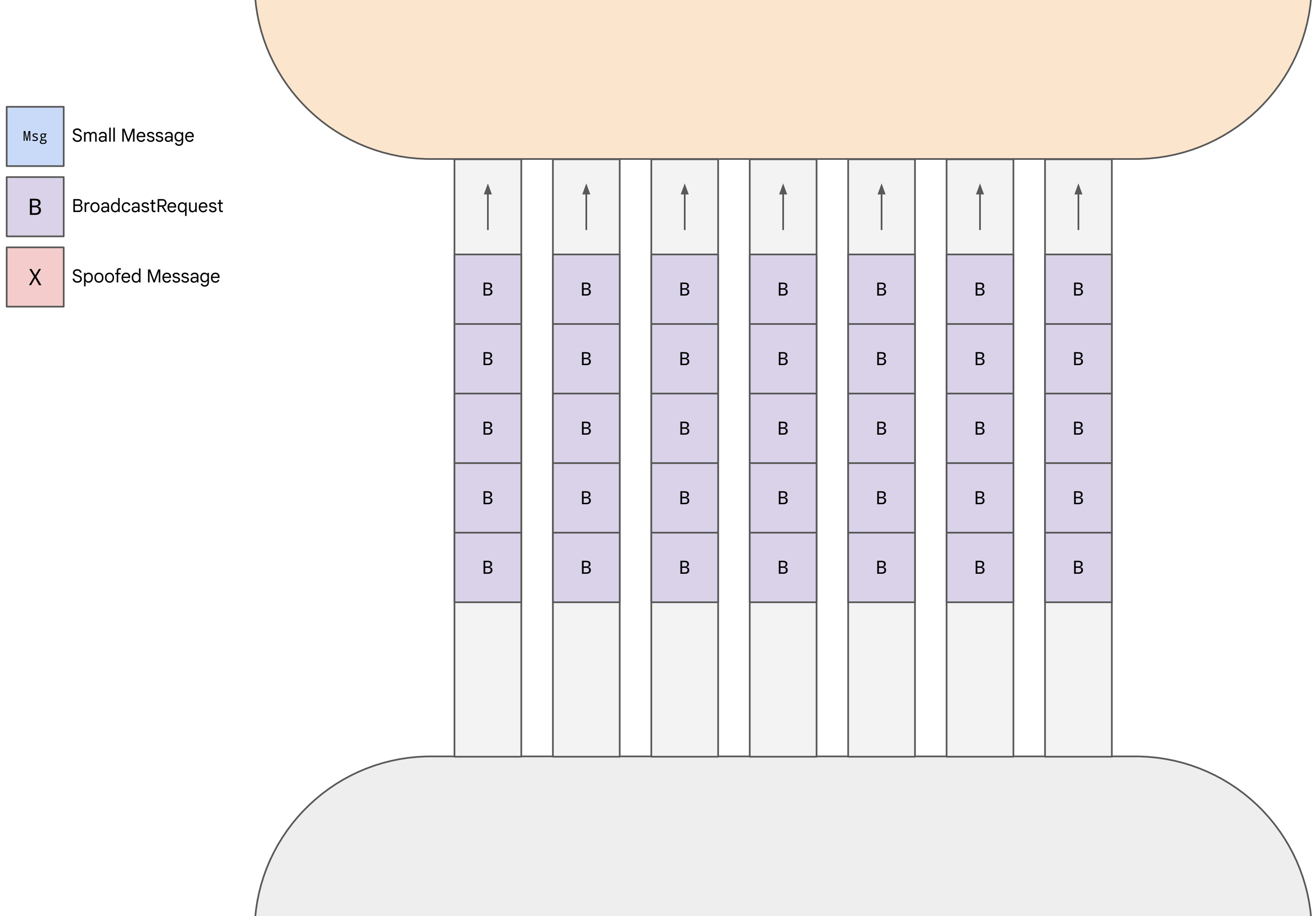
Network Process

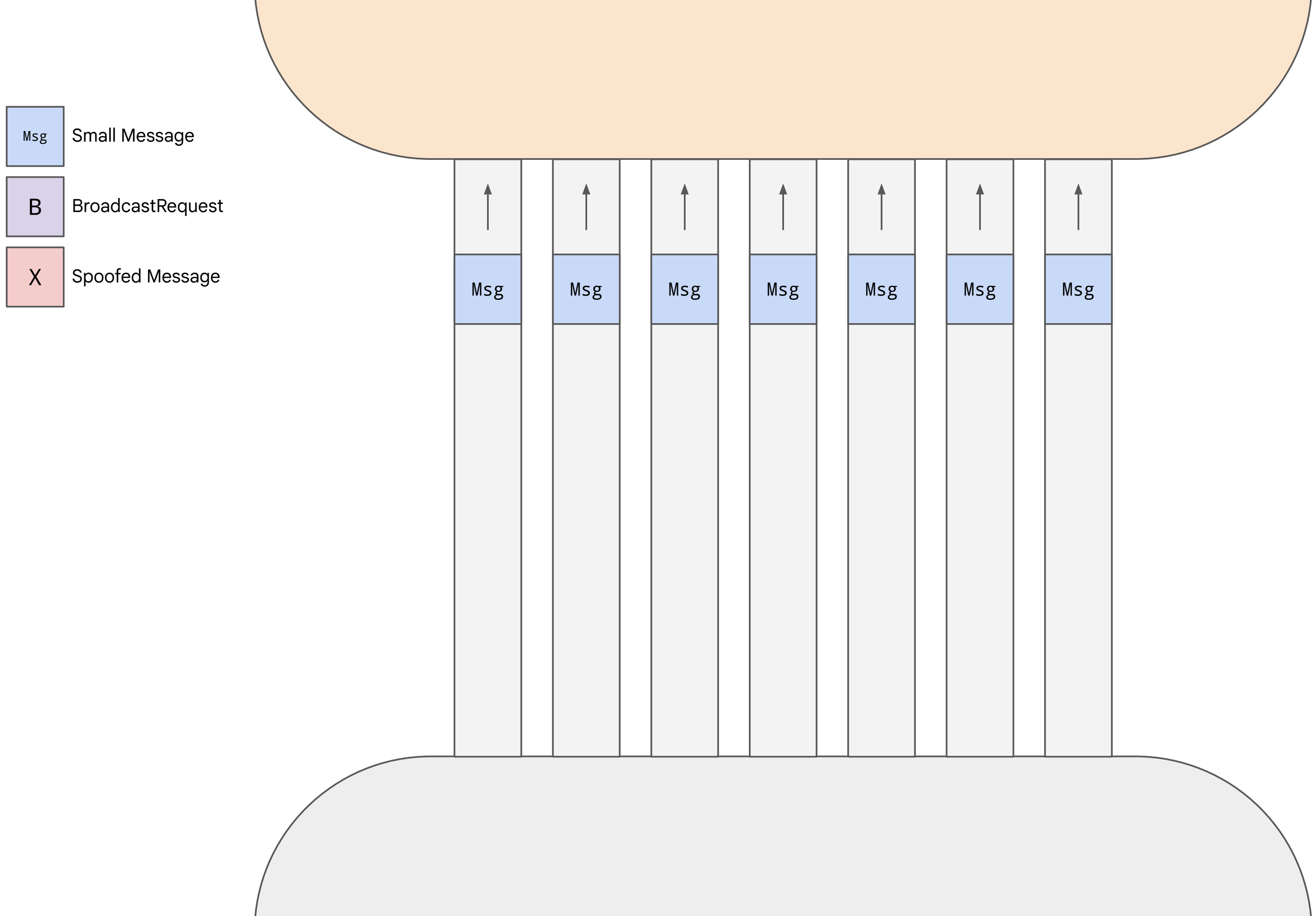


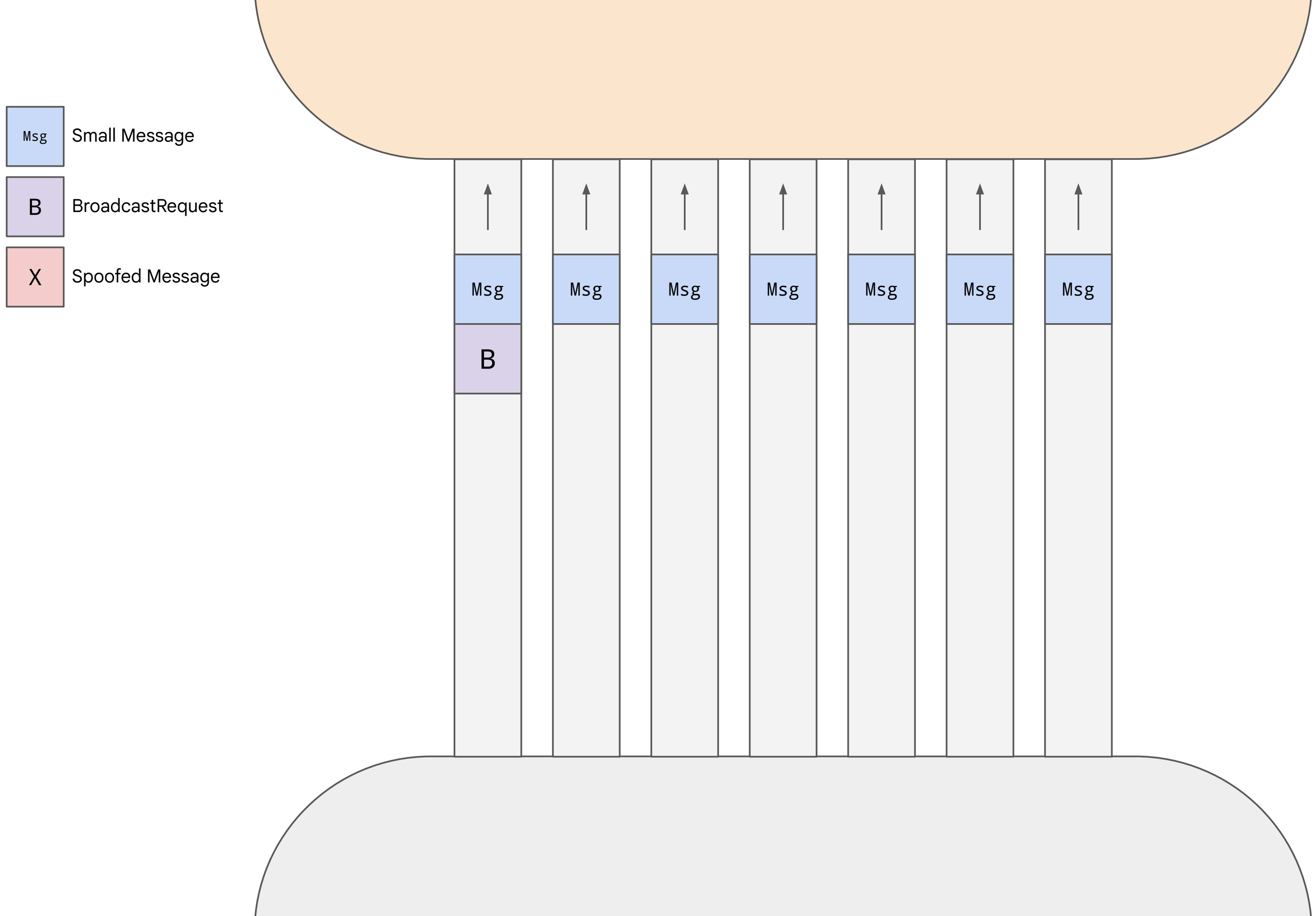

Network Process
DoS
Read Spoofed Message
Destroy all ports with peer
Enqueue OnError Handlers
Process Spoofed Message
DoS
OnError Tasks

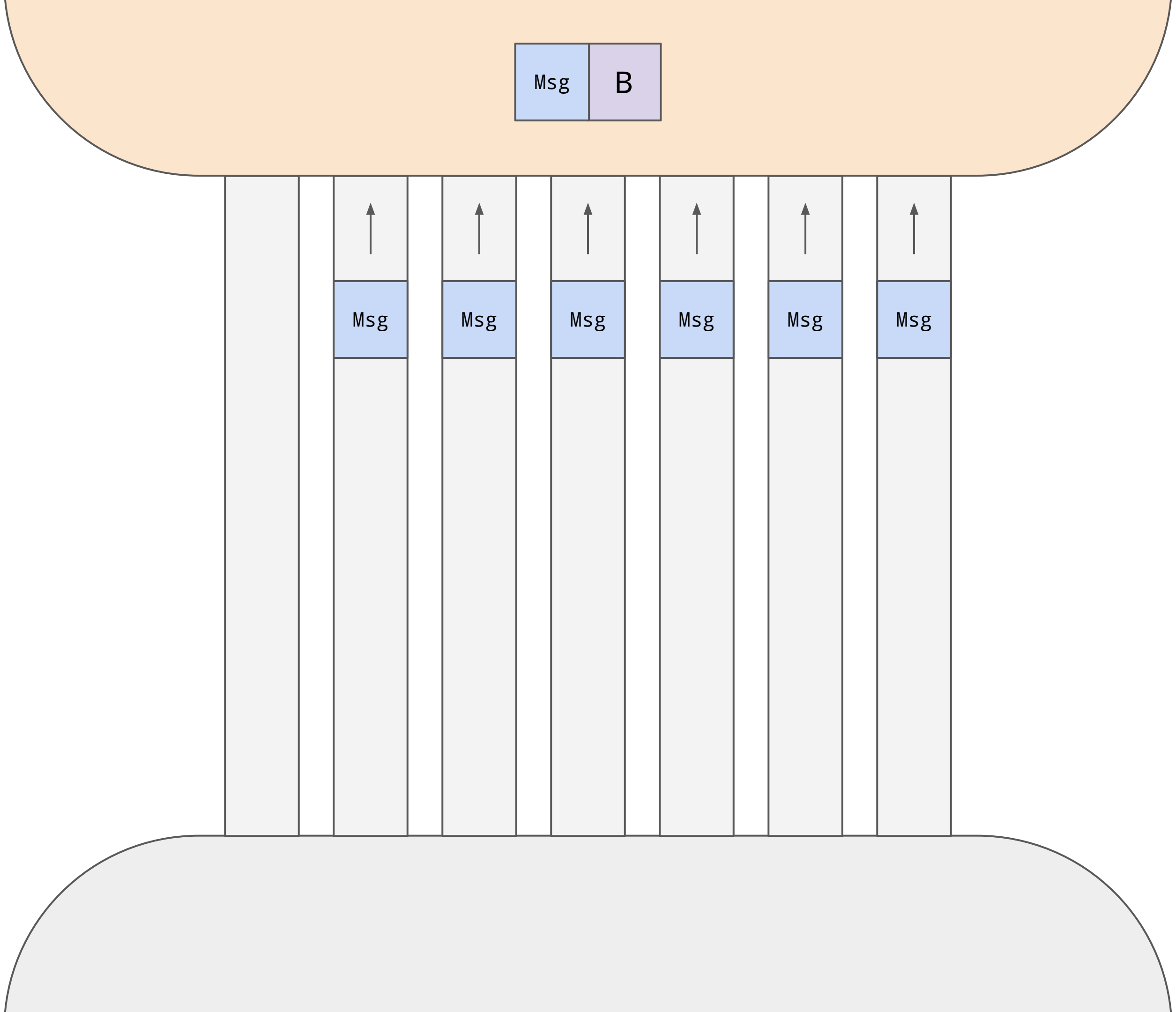



Msg
Small Message
B
BroadcastRequest
X
Spoofed Message

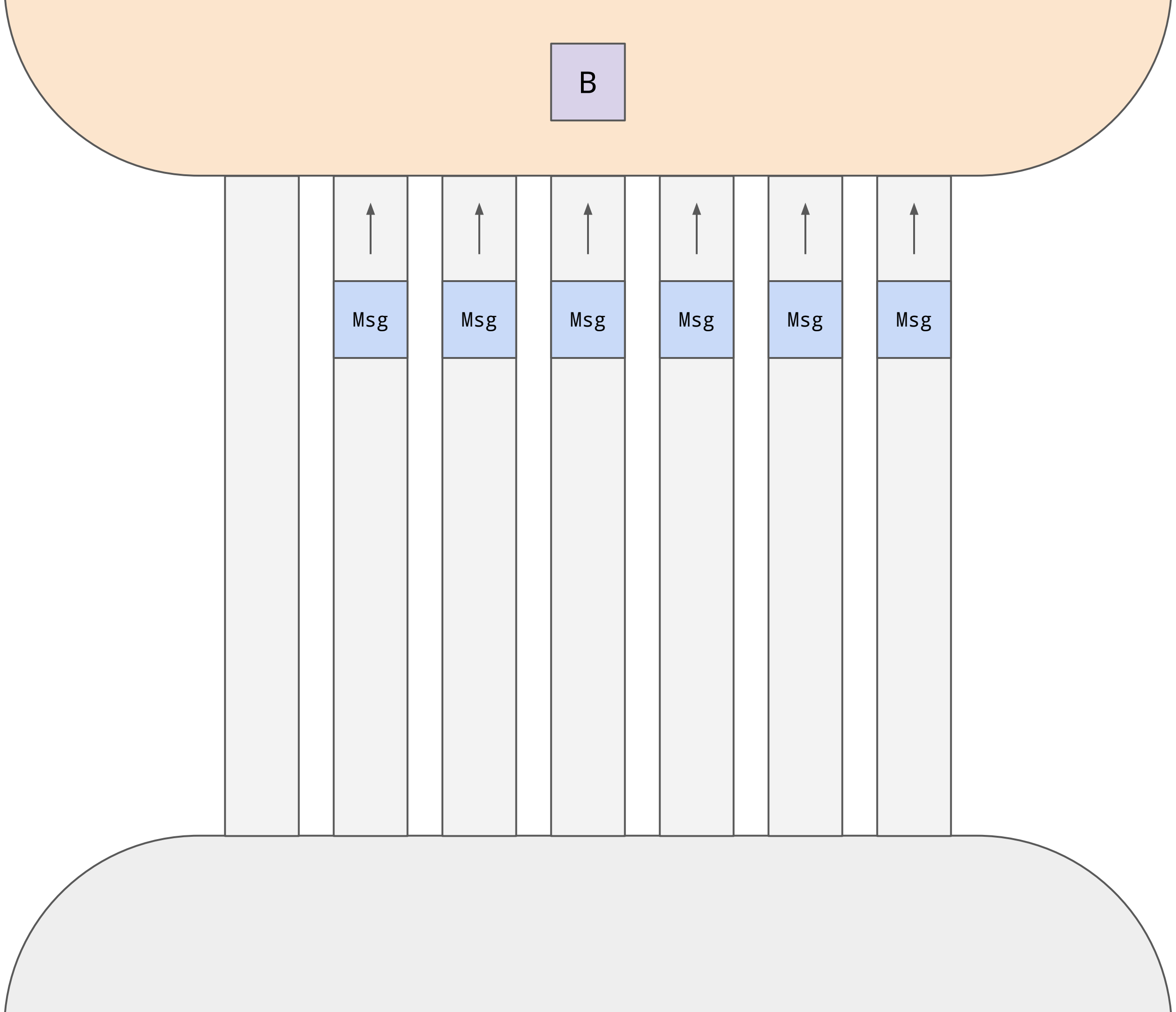

Msg
Small Message
B
BroadcastRequest
X
Spoofed Message
B


Msg
Small Message
B
BroadcastRequest
X
Spoofed Message
Msg
Msg
Msg
Msg
Msg
Msg
Msg

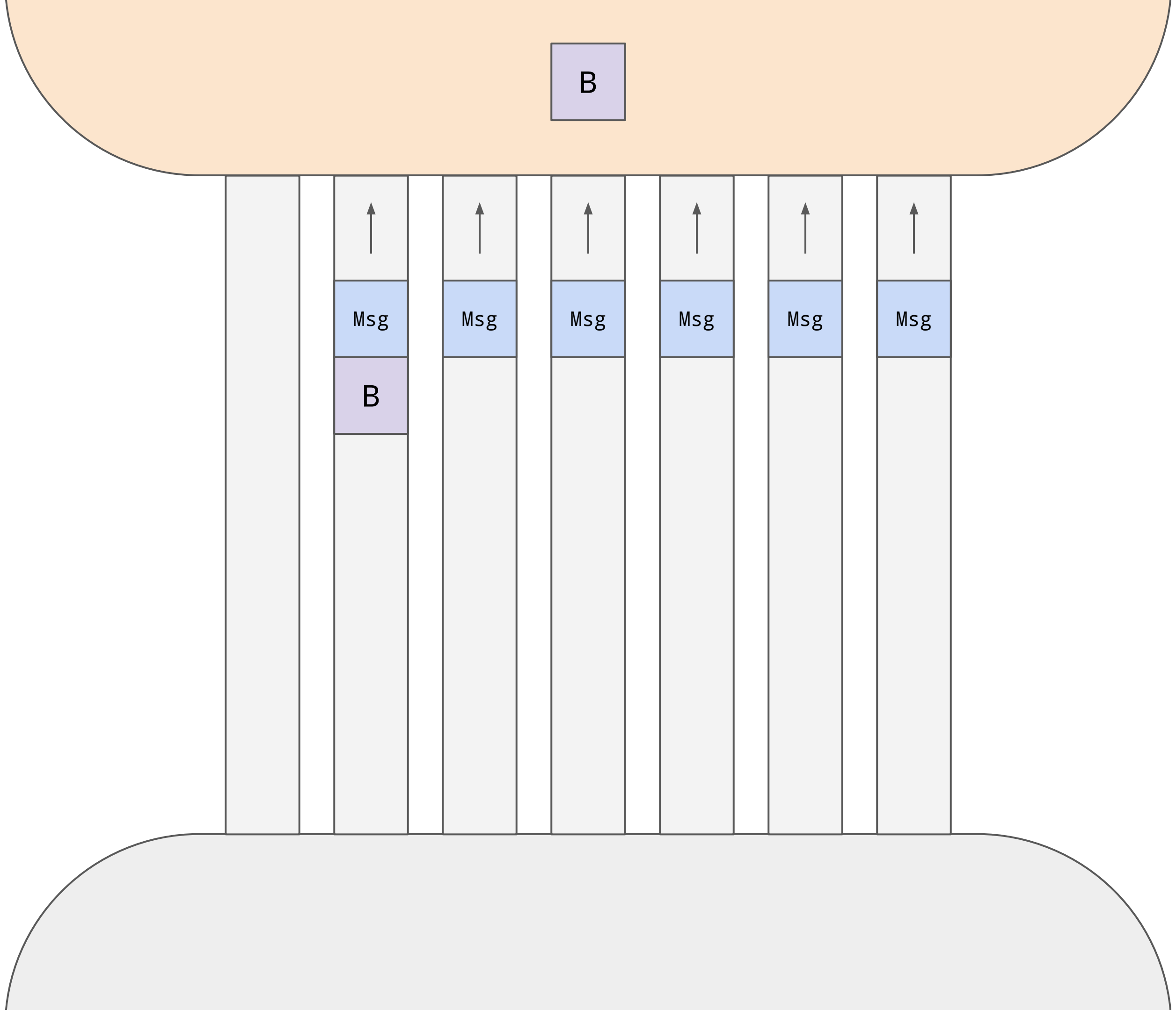

Msg
Small Message
B
BroadcastRequest
X
Spoofed Message
Msg
Msg
Msg
Msg
Msg
Msg
Msg
B



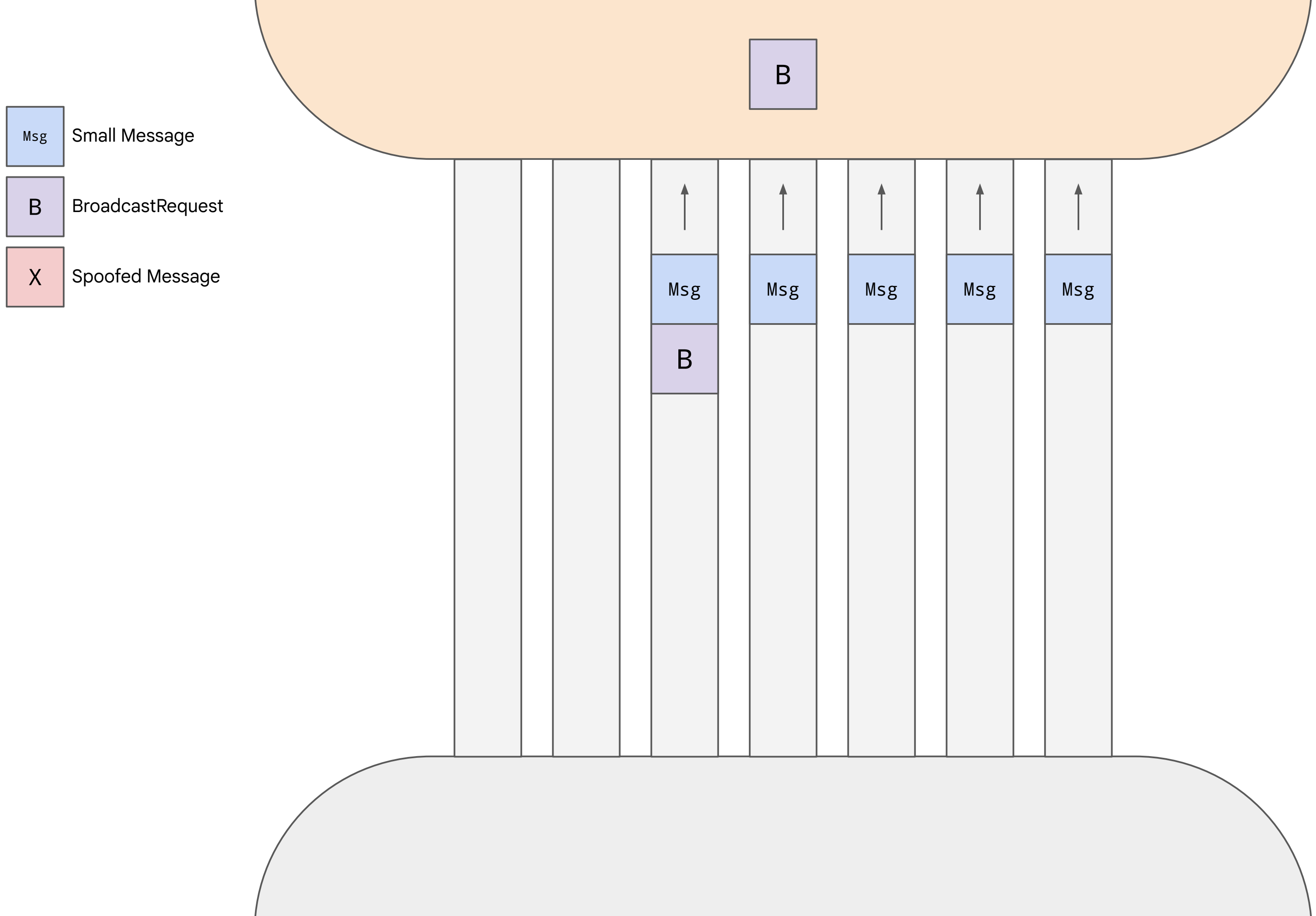
Msg	Small Message
B	BroadcastRequest
X	Spoofed Message

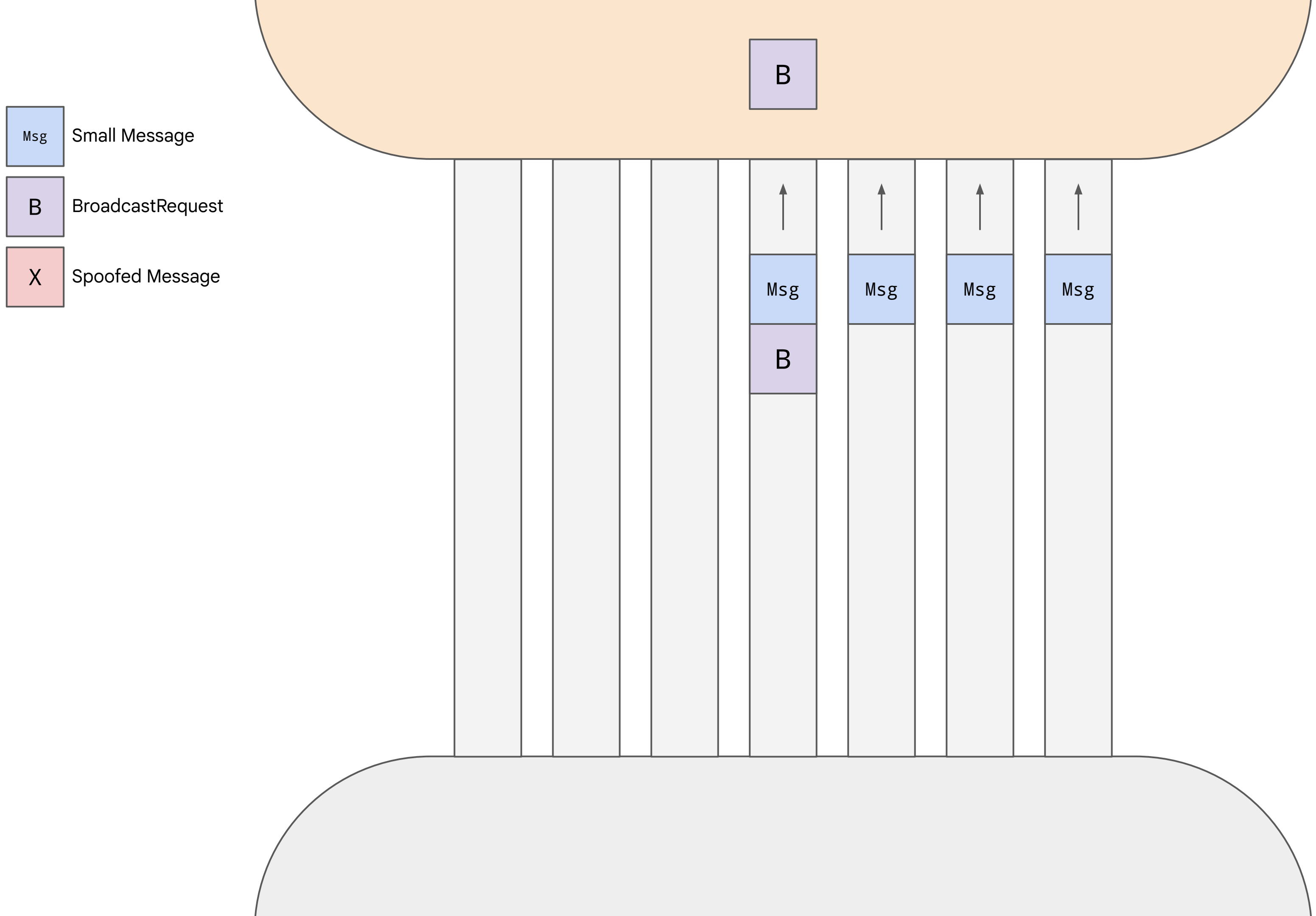


Msg	Small Message
B	BroadcastRequest
X	Spoofed Message

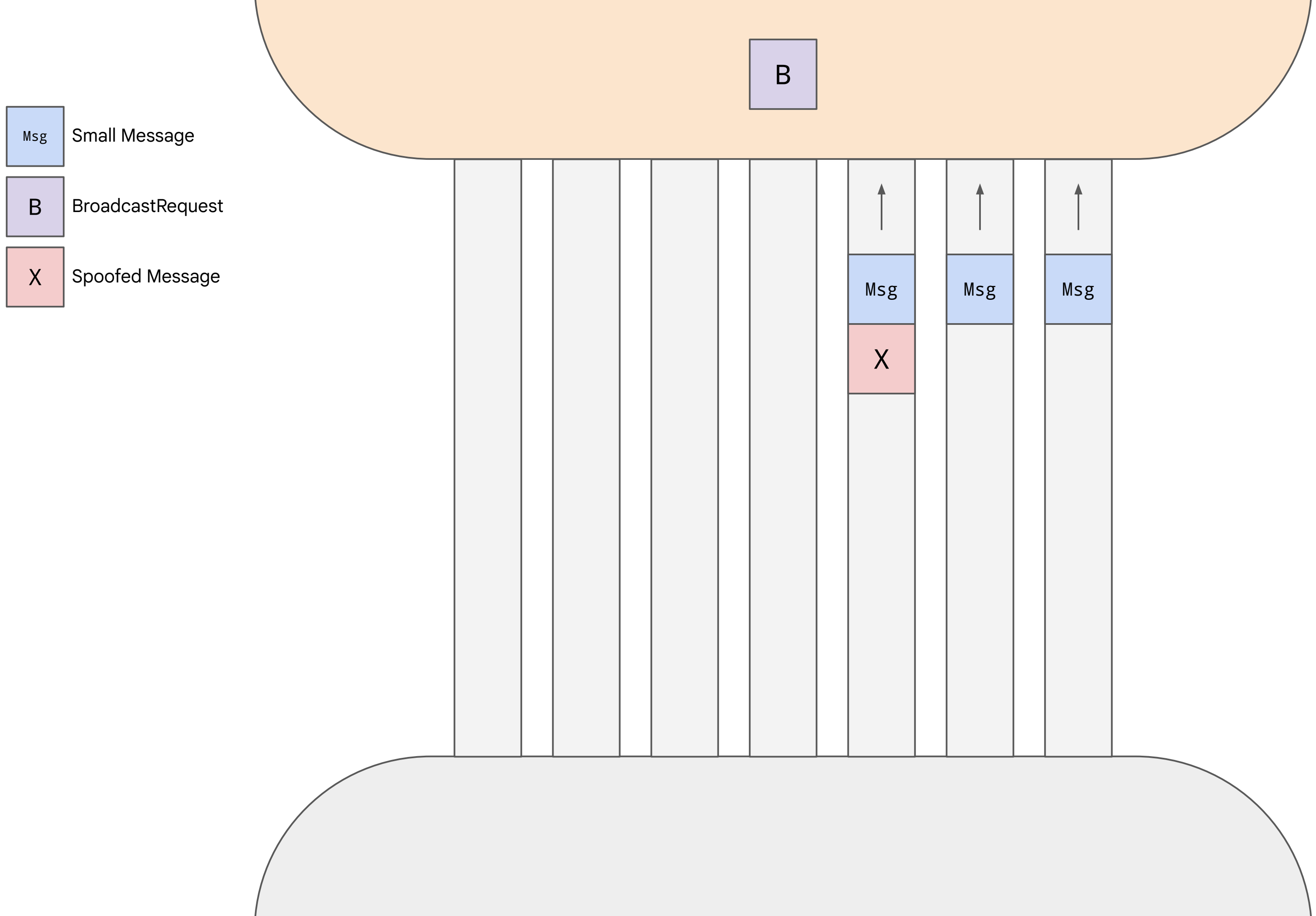


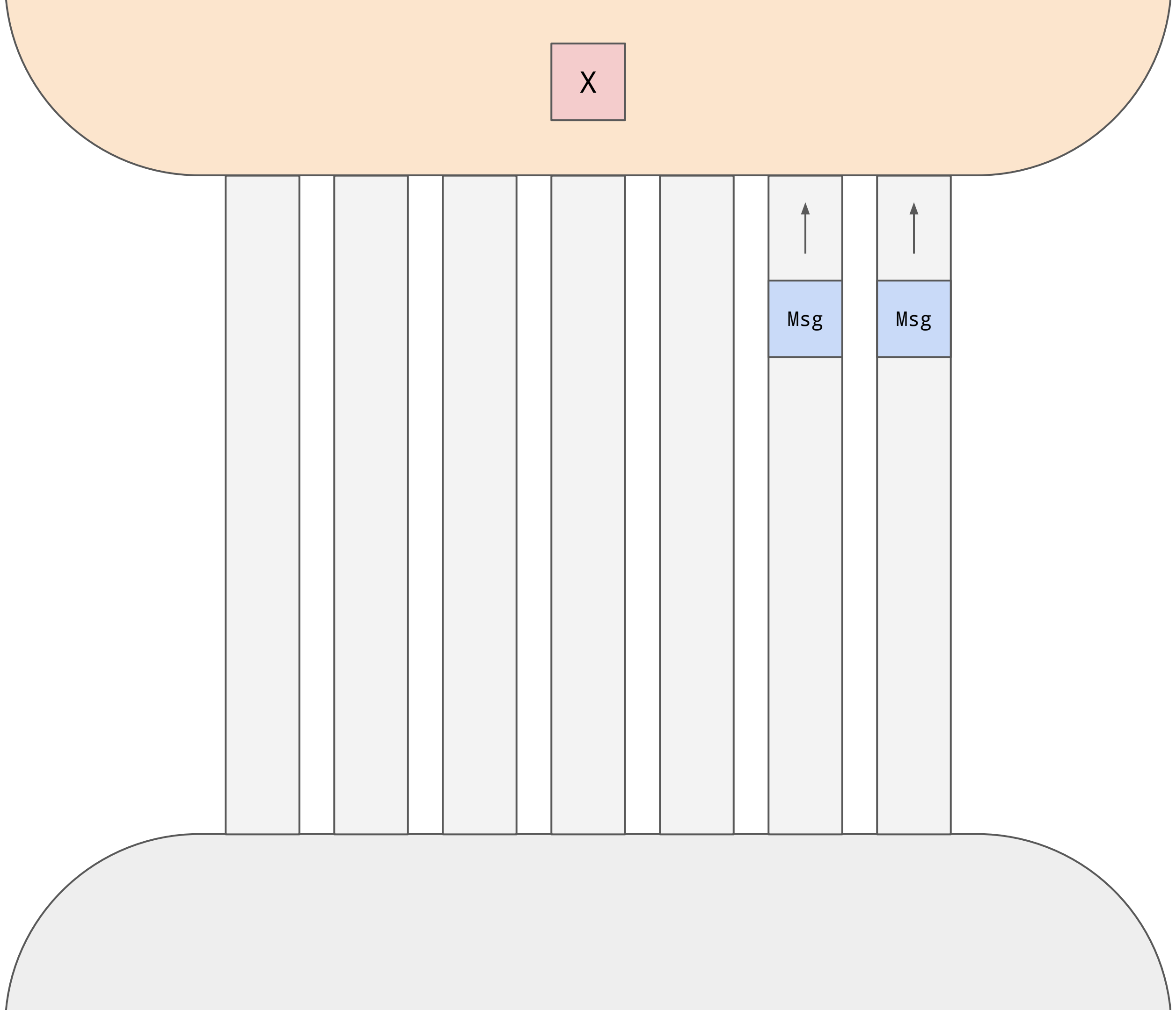
Msg	Small Message
B	BroadcastRequest
X	Spoofed Message


Msg
Small Message
B
BroadcastRequest
X
Spoofed Message
B
Msg
Msg
Msg
Msg
Msg
B

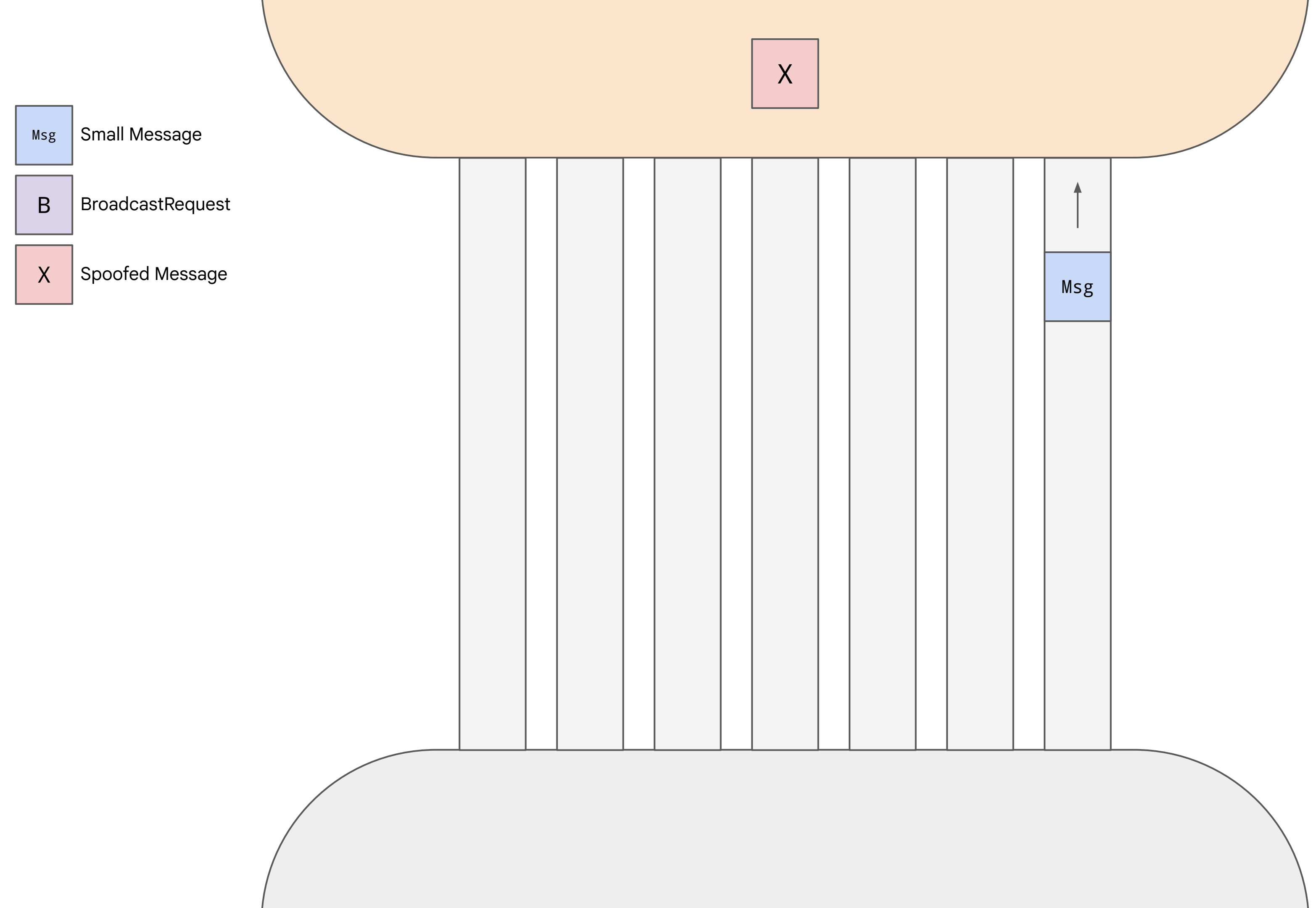


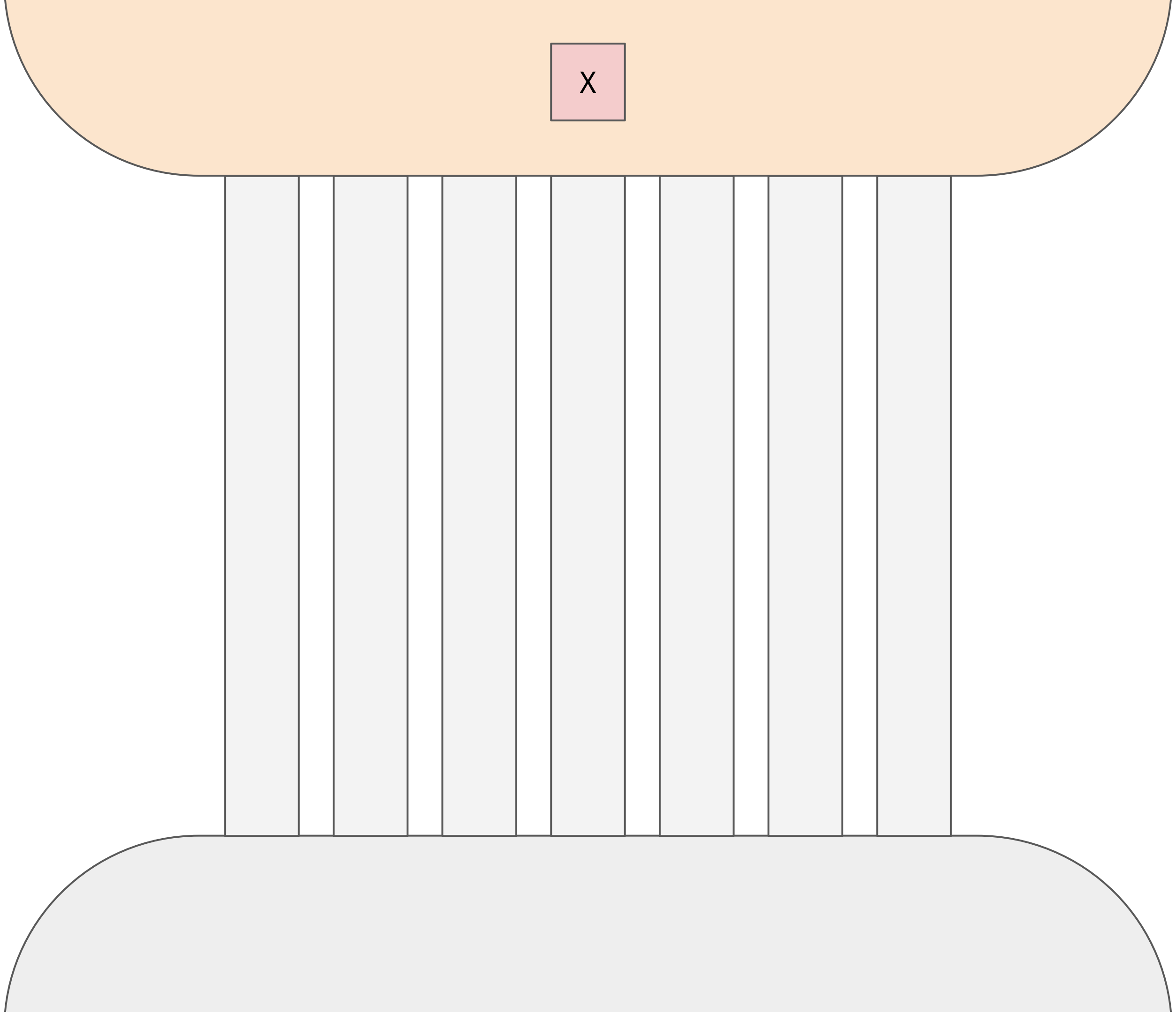
Msg
Small Message
B
BroadcastRequest
X
Spoofed Message
B
Msg
Msg
Msg
Msg
B


Msg
Small Message
B
BroadcastRequest
X
Spoofed Message
B
Msg
Msg
Msg
X

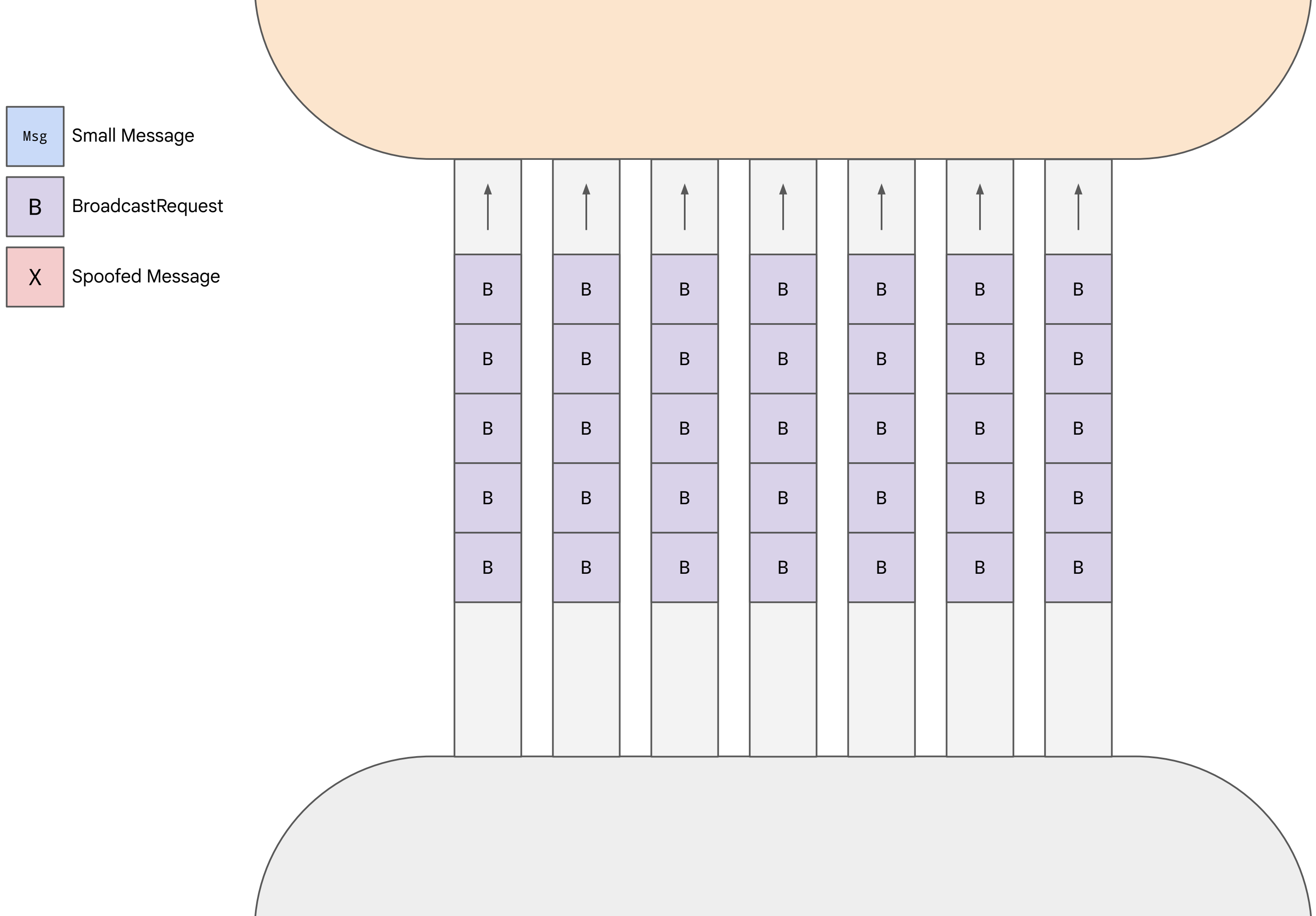


Msg	Small Message
B	BroadcastRequest
X	Spoofed Message


Msg
Small Message
B
BroadcastRequest
X
Spoofed Message
X
Msg



Msg	Small Message
B	BroadcastRequest
X	Spoofed Message


Msg Small Message
B BroadcastRequest
X Spoofed Message
B

HAL

They're coming in from remote nodes. They're going after the kernel!

```
// A control interface the browser uses to drive the behavior of all types of
// Content child processes.
interface ChildProcess {
  // Requests that the process bind a receiving pipe targeting the service
  // interface named by |receiver|.
  BindServiceInterface(mojo_base.mojom.GenericPendingReceiver receiver);
};
```

```
void RegisterMainThreadServices(mojo::ServiceFactory& services) {
  services.Add(RunAuctionWorkletService);
  services.Add(RunAudio);

  services.Add(RunDataDecoder);
  services.Add(RunStorageService);
  services.Add(RunTracing);
  services.Add(RunVideoCapture);

```

PHREAK

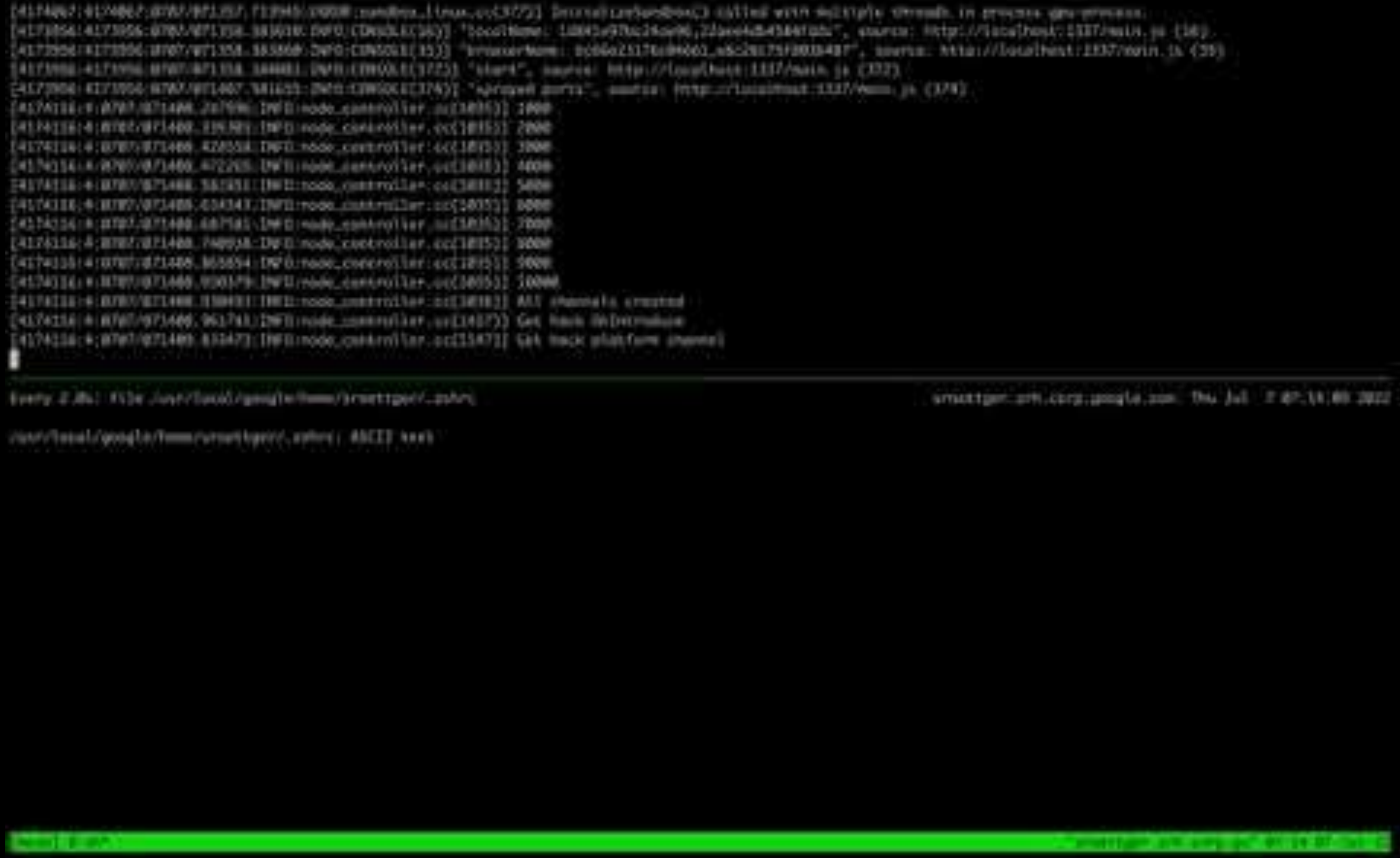
Yo, check this out guys, this is insanely great. It's got a JavaScript engine!

:(

```
struct HttpAuthStaticParams {
  // List of supported auth schemes. Unrecognized schemes are ignored.
  // The default value of this field (an empty list) does not match default
  // behavior of NetworkService when no HttpAuthStaticParams is specified.
  array<string> supported_schemes;

  // File name the GSSAPI library to load. Only supported on platforms where an
  // external GSSAPI library is necessary for Kerberos/SPNEGO support. See the
  // |use_external_gssapi| variable definition in //net/BUILD.gn for details on
  // platforms where this setting is applicable.
  string gssapi_library_name;
};
```

:(

```
INSERT INTO cookies VALUES('\r\ncalc.exe\r\n')
```



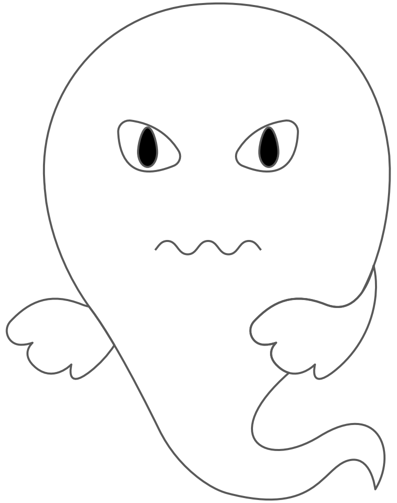
```
[4174067:4174067:0707/071357.713945:ERROR:sandbox_linux.cc(377)] InitializeSandbox() called with multiple threads in process gpu-process.
[4173956:4173956:0707/071358.383010:INFO:CONSOLE(16)] "localName: [illegible]", source: http://localhost:1337/main.js (16)
[4173956:4173956:0707/071358.383060:INFO:CONSOLE(35)] "browserName: [illegible]", source: http://localhost:1337/main.js (35)
[4173956:4173956:0707/071358.384081:INFO:CONSOLE(372)] "start", source: http://localhost:1337/main.js (372)
[4173956:4173956:0707/071407.581655:INFO:CONSOLE(376)] "sprayed ports", source: http://localhost:1337/main.js (376)
[4174116:4:0707/071408.287996:INFO:node_controller.cc(1035)] 1000
[4174116:4:0707/071408.335301:INFO:node_controller.cc(1035)] 2000
[4174116:4:0707/071408.428558:INFO:node_controller.cc(1035)] 3000
[4174116:4:0707/071408.472265:INFO:node_controller.cc(1035)] 4000
[4174116:4:0707/071408.561651:INFO:node_controller.cc(1035)] 5000
[4174116:4:0707/071408.634347:INFO:node_controller.cc(1035)] 6000
[4174116:4:0707/071408.687581:INFO:node_controller.cc(1035)] 7000
[4174116:4:0707/071408.740938:INFO:node_controller.cc(1035)] 8000
[4174116:4:0707/071408.865654:INFO:node_controller.cc(1035)] 9000
[4174116:4:0707/071408.930179:INFO:node_controller.cc(1035)] 10000
[4174116:4:0707/071408.930493:INFO:node_controller.cc(1038)] All channels created
[4174116:4:0707/071408.961743:INFO:node_controller.cc(1437)] Got back OnIntroduce
[4174116:4:0707/071409.833473:INFO:node_controller.cc(1547)] Got back platform channel
█

Every 2.0s: file /usr/local/google/home/[illegible]/.zshrc                    [illegible].corp.google.com: Thu Jul  7 07:14:09 2022

/usr/local/google/home/[illegible]/.zshrc: ASCII text
```

[illegible]

Takeaway

Target-specific knowledge can be crucial

- Find bugs that fuzzers will never trigger
- Turn impossible bugs exploitable



Got stuck? Watch Hackers!