digium
The Asterisk Company

Architecture of Asterisk 13

Matt Jordan
@mattcjordan
Director of Technology, Digium

Why this talk?

- Asterisk 12
  - New SIP stack
  - New APIs
  - New Core

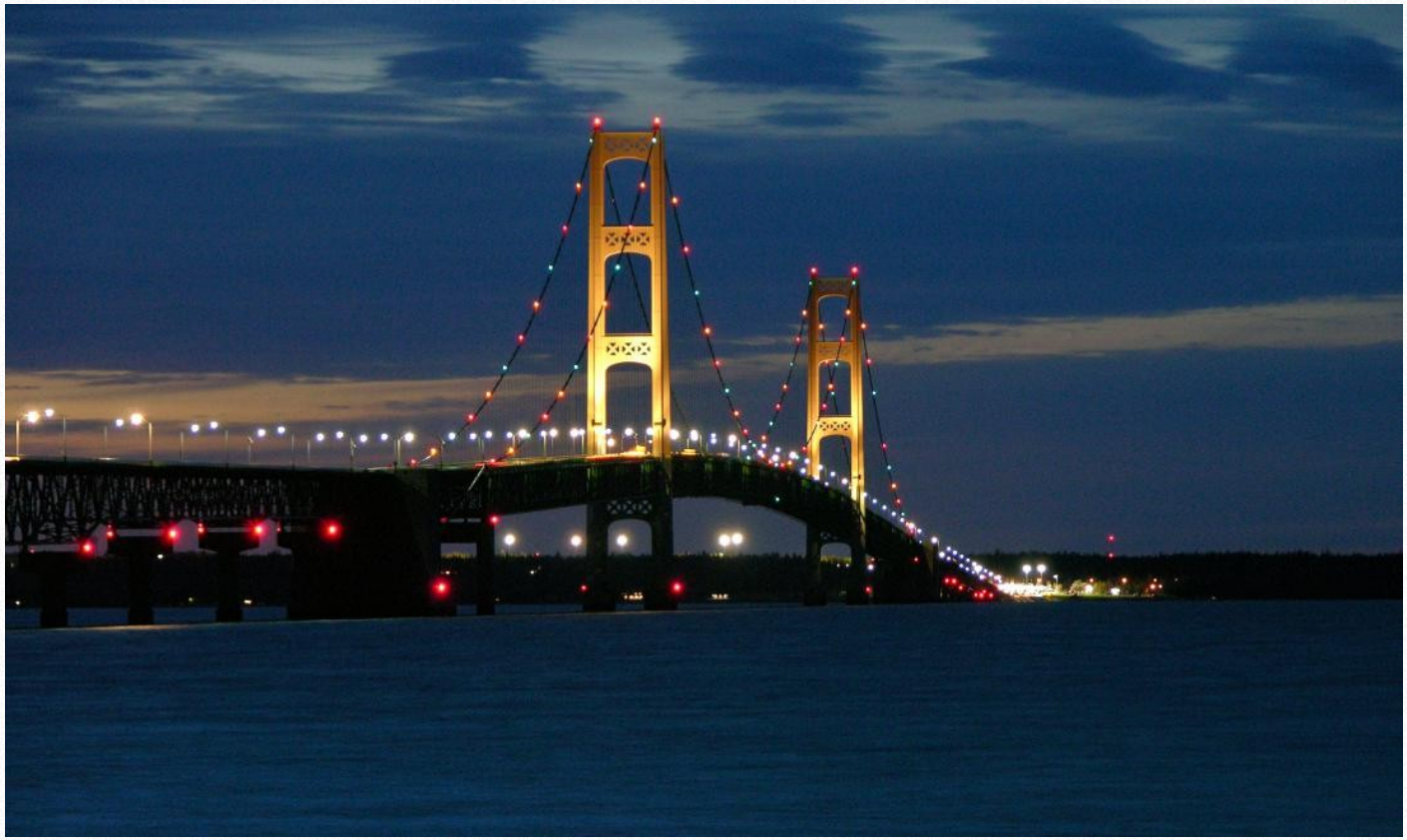
Why this talk?

- Asterisk 12
  - New SIP stack
  - New APIs
  - New Core
- Kamailio World:
  - 2014 - Asterisk's PJSIP stack
  - 2015 - Asterisk APIs/ARI

Why this talk?

- Asterisk 12
  - New SIP stack
  - New APIs
  - New Core
- Kamailio World:
  - 2014 - Asterisk's PJSIP stack
  - 2015 - Asterisk APIs/ARI
- Let's talk about C APIs!

Bridging!



The Problem

Or: How to justify to your employers spending 10 man years on something you hope your end users never notice

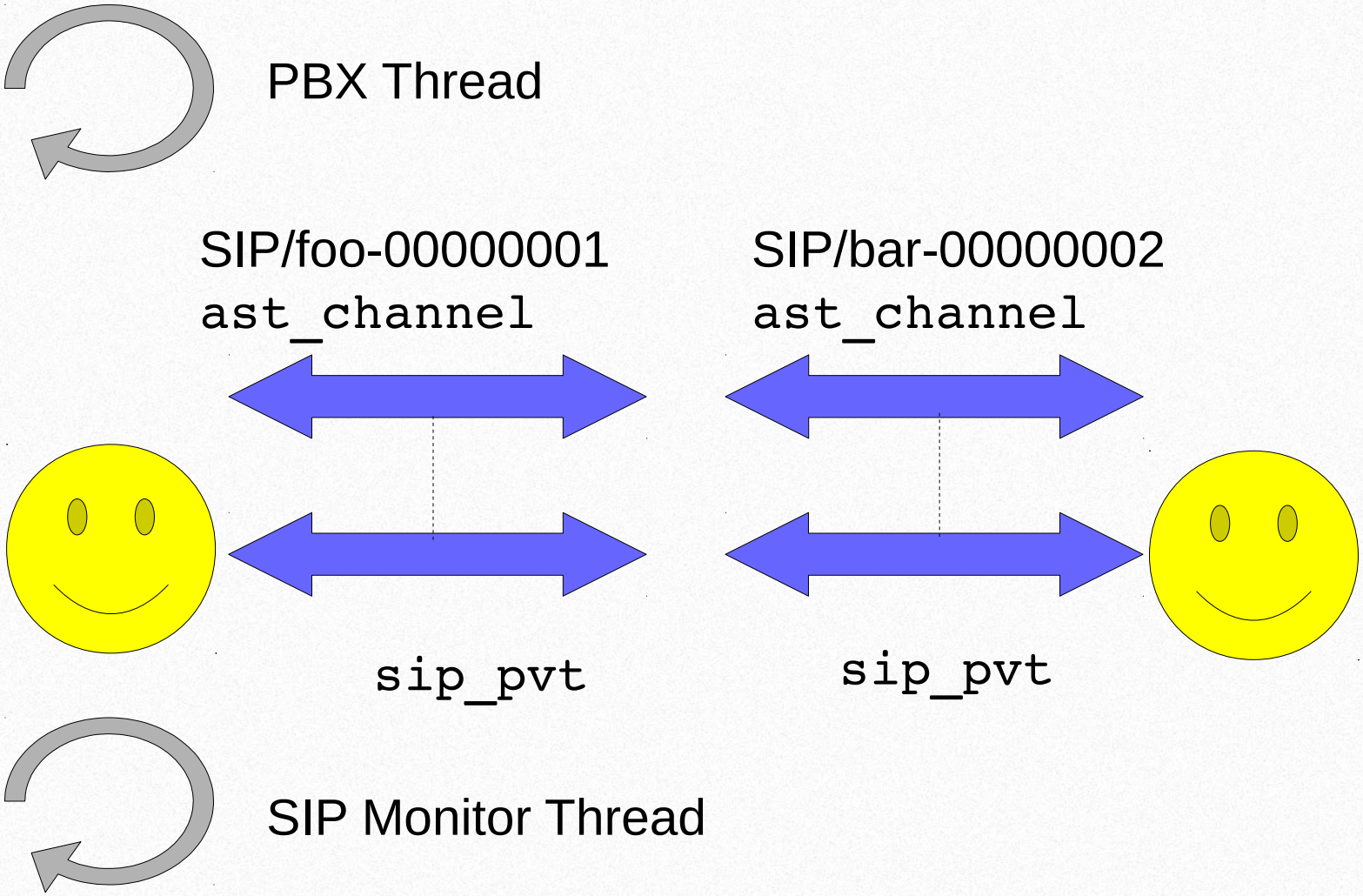
The Problem

- Asterisk
  - Parallelism achieved through multithreading
  - “Conservative” threading model
    - One thread per “call”
    - Thread services both inbound and outbound channels

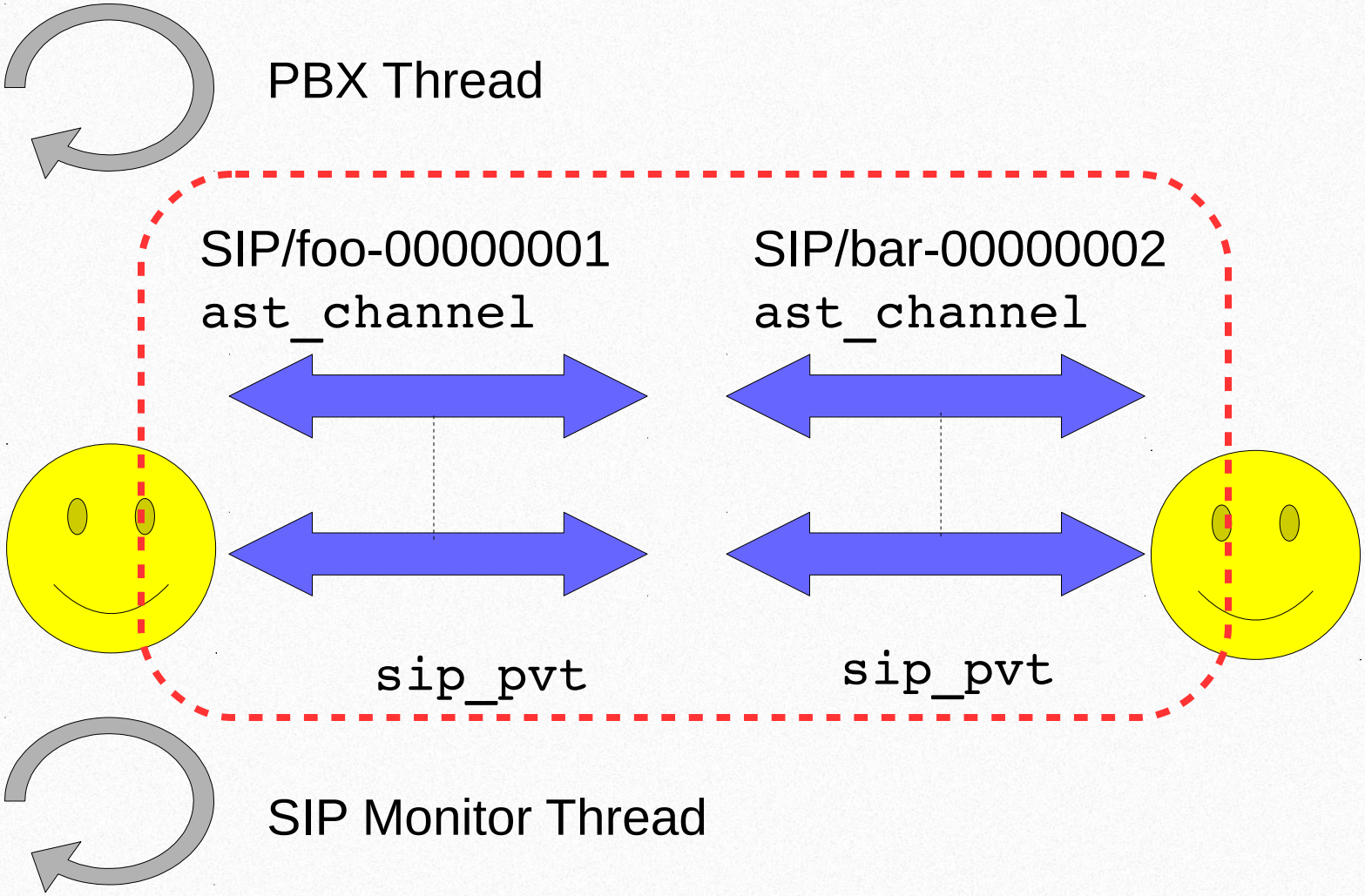
The Problem

- Asterisk
  - Parallelism achieved through multithreading
  - “Conservative” threading model
    - One thread per “call”
    - Thread services both inbound and outbound channels
- Problem: How do you move channels out of a call?
  - Transfers
  - Externally initiated redirects
  - Parking
  - Call pickup

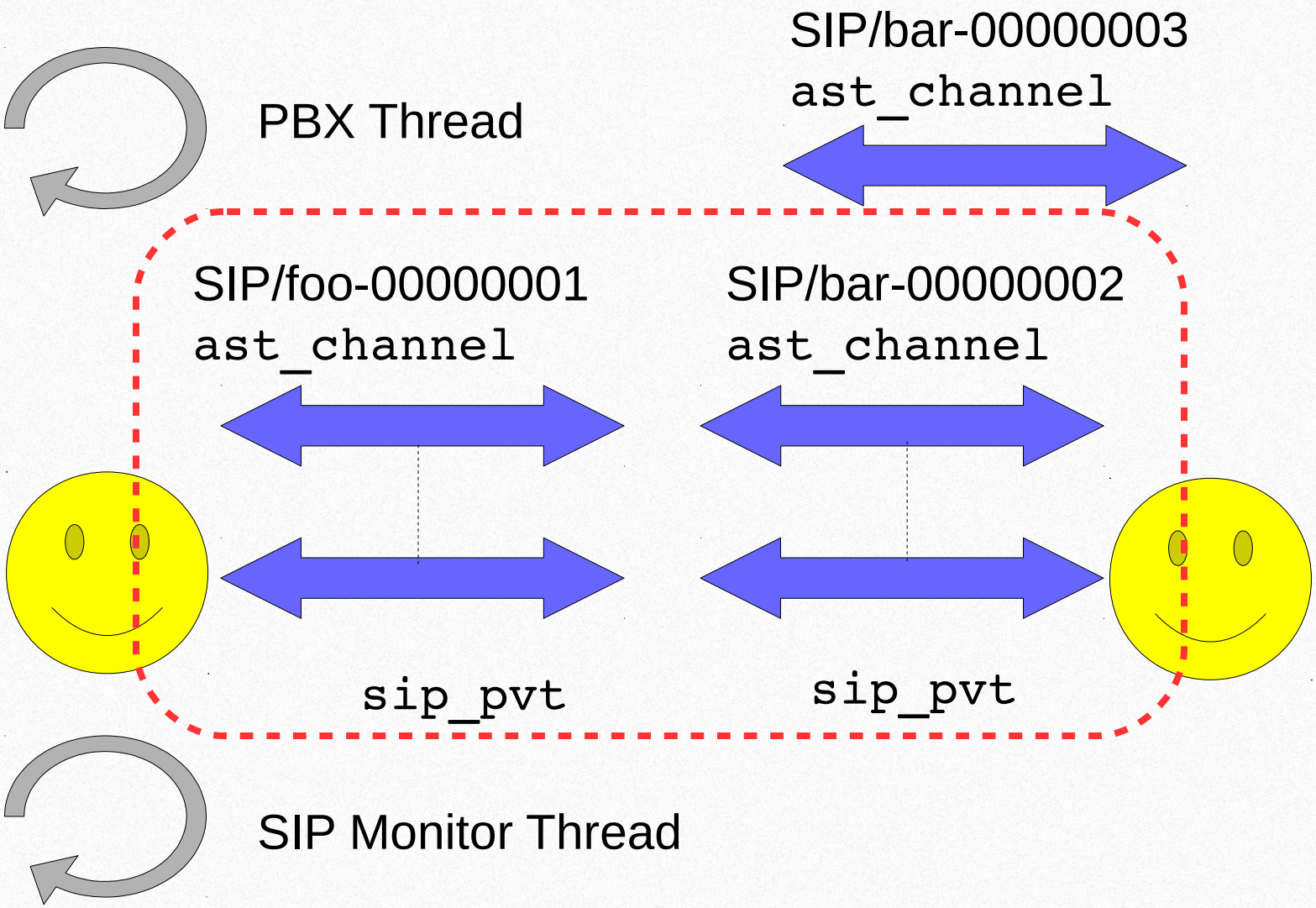
“Solution”: Masquerades!



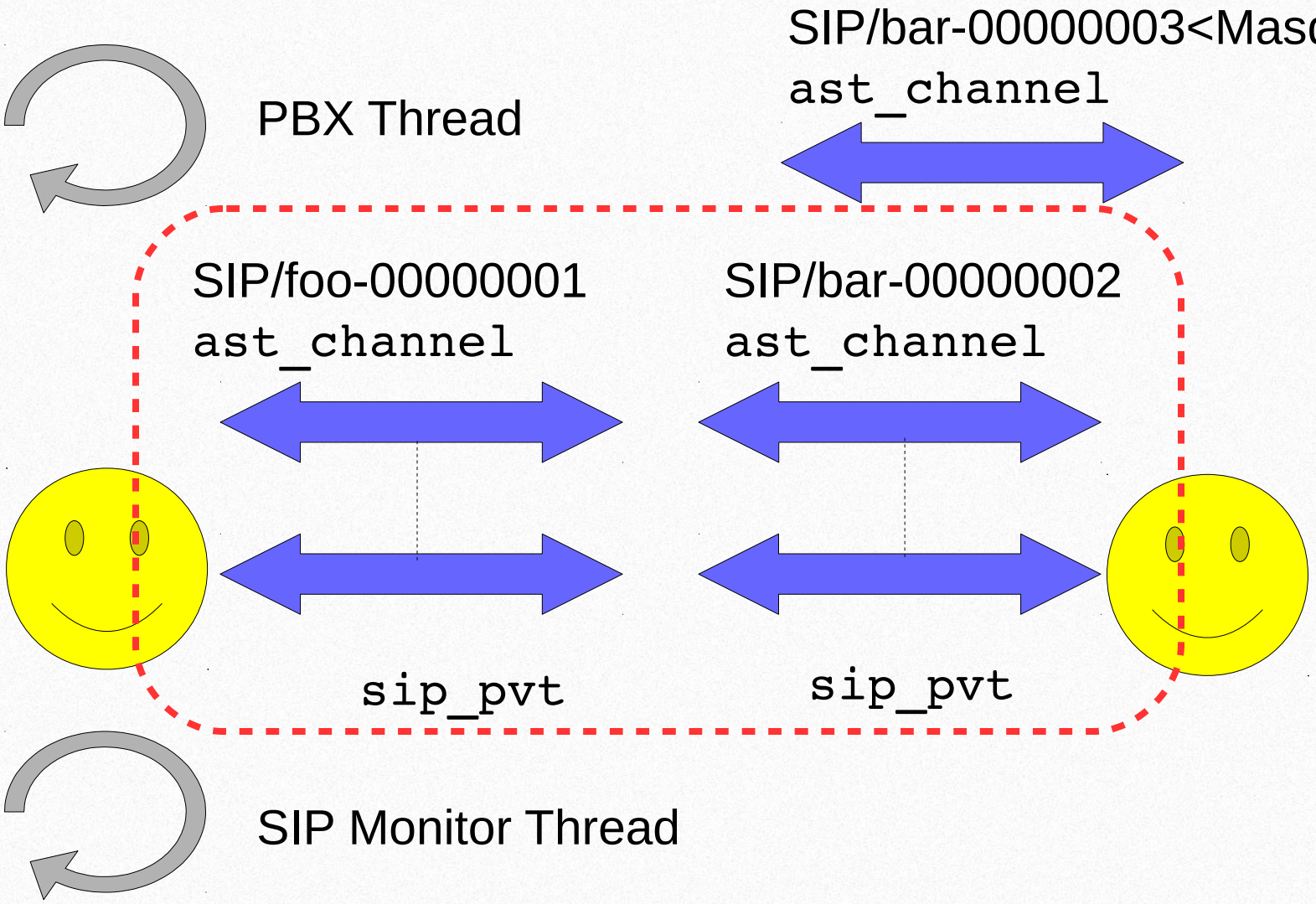
“Solution”: Masquerades!



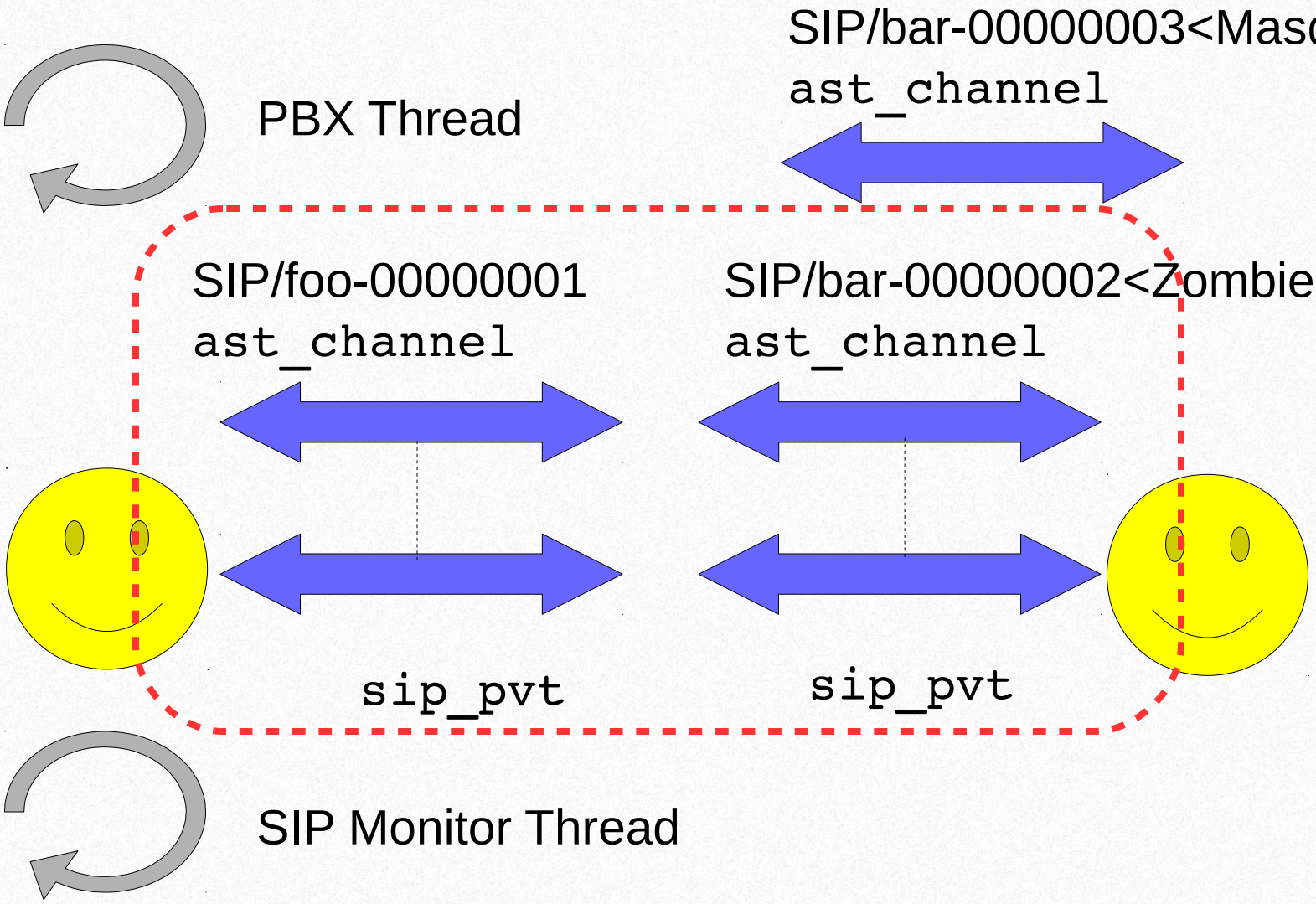
“Solution”: Masquerades!



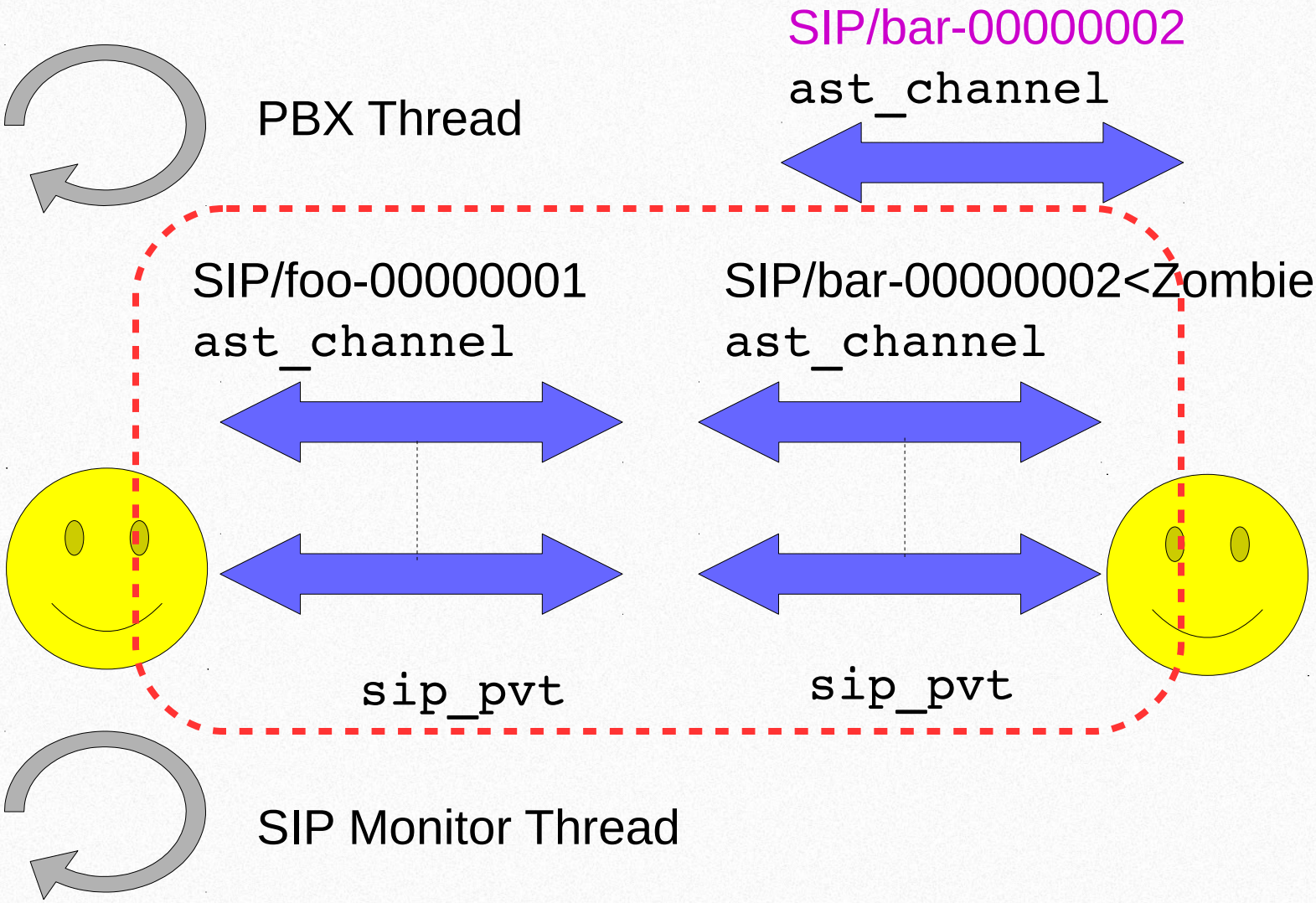
“Solution”: Masquerades!



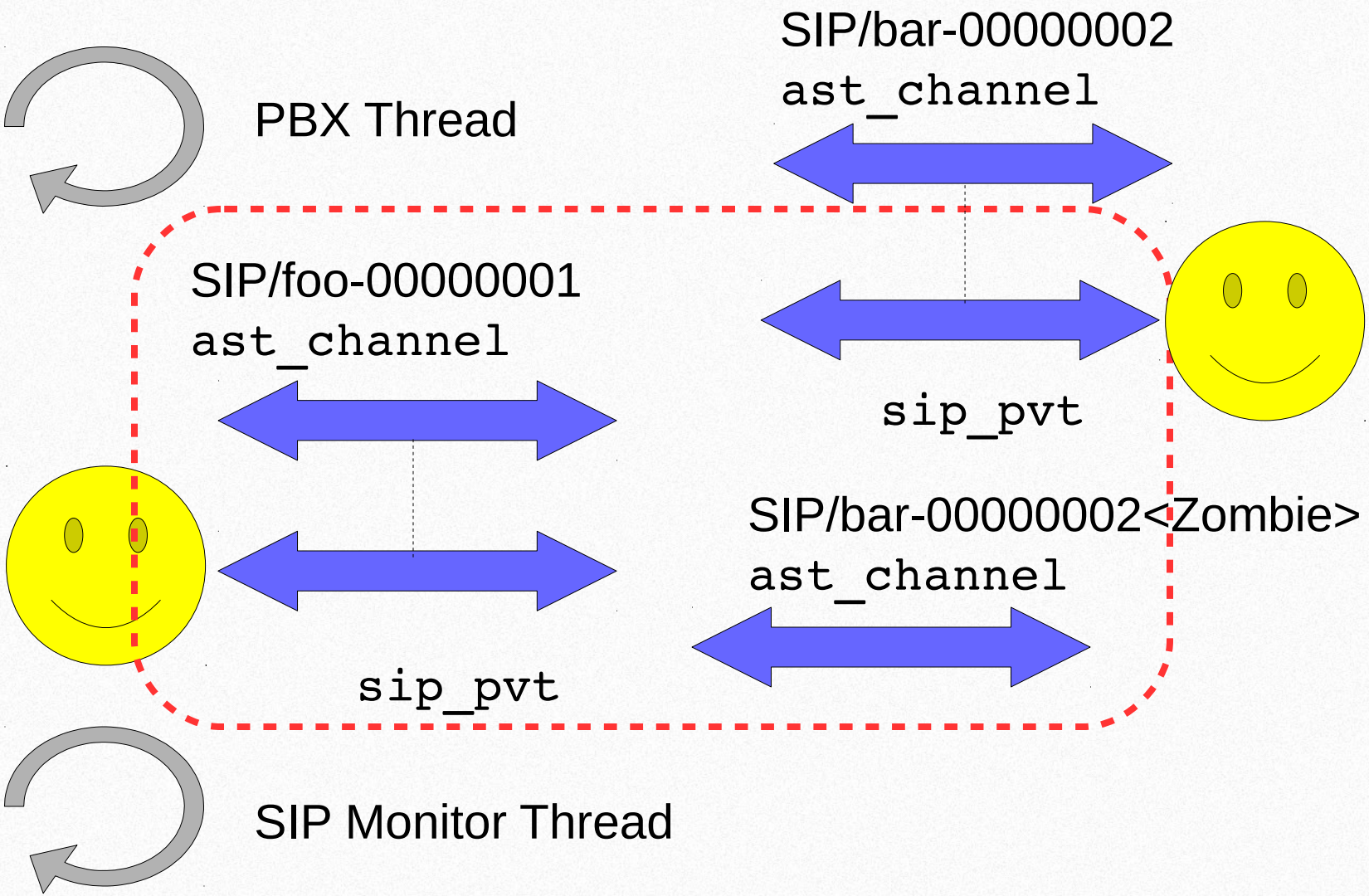
“Solution”: Masquerades!



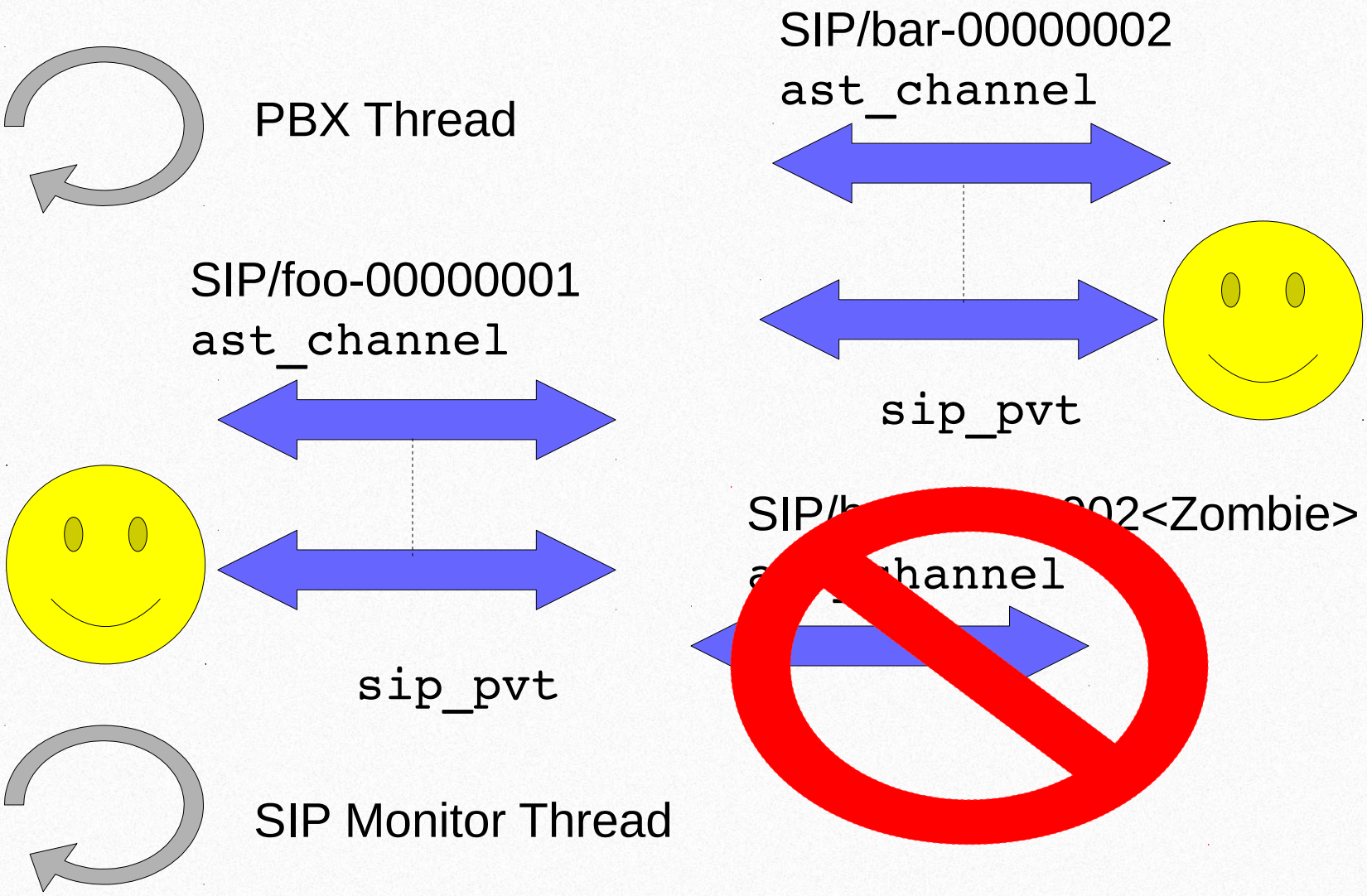
“Solution”: Masquerades!



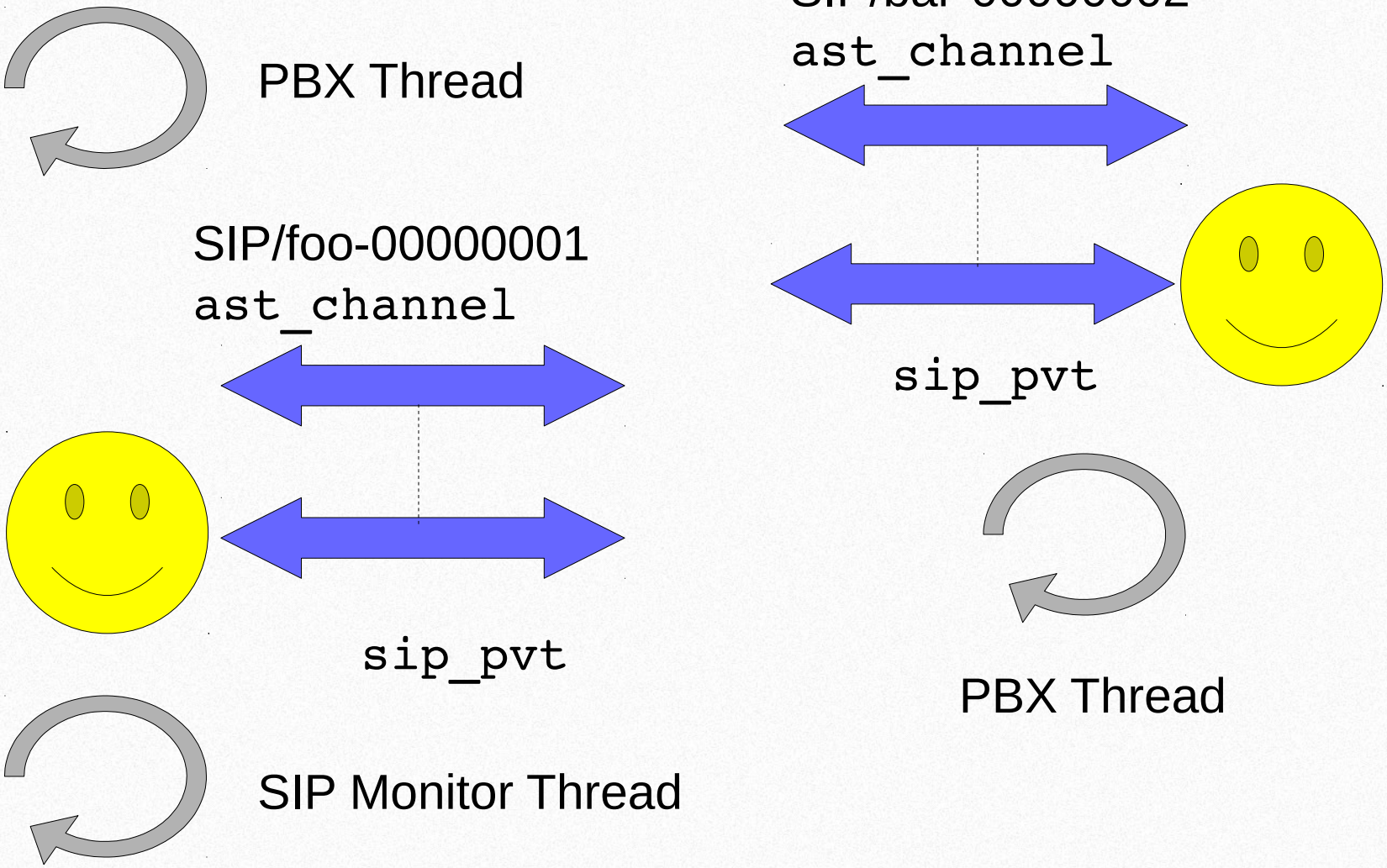
“Solution”: Masquerades!



“Solution”: Masquerades!



“Solution”: Masquerades!




THIS IS FINE.

Masquerade Events

```
Event: NewChannel
Channel: SIP/bar-00000003

Event: Rename
Channel: SIP/bar-00000003
Newname: SIP/bar-00000003<Masq>

Event: Rename
Channel: SIP/bar-00000002
Newname: SIP/bar-00000002<Zombie>

Event: Rename
Channel: SIP/bar-00000003<Masq>
Newname: SIP/bar-00000002

Event: Masquerade
Original: SIP/bar-00000002
Clone: SIP/bar-00000002<Zombie>
```


I'M OKAY WITH THE EVENTS THAT ARE UNFOLDING CURRENTLY.

Problem: “Bridging”

```
features::
ast_do_bridge
```

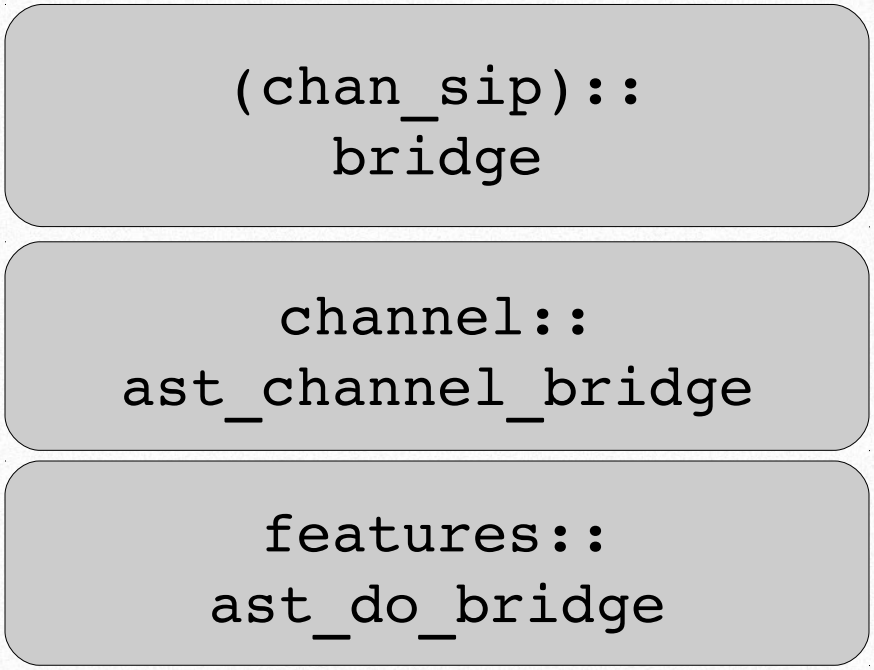
Problem: “Bridging”

digium
The Asterisk Company

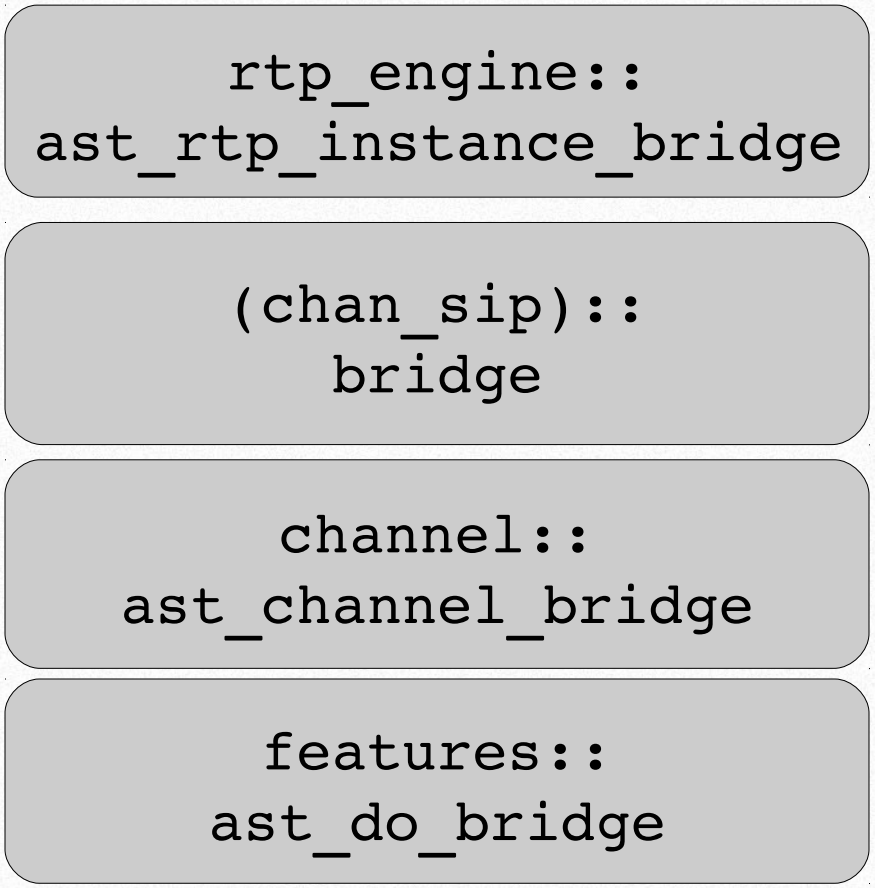
```
channel::
ast_channel_bridge
```

```
features::
ast_do_bridge
```

Problem: “Bridging”



Problem: “Bridging”



Problem: “Bridging”

digium
The Asterisk Company

```
(res_rtp_asterisk)::
ast_rtp_local_bridge
```

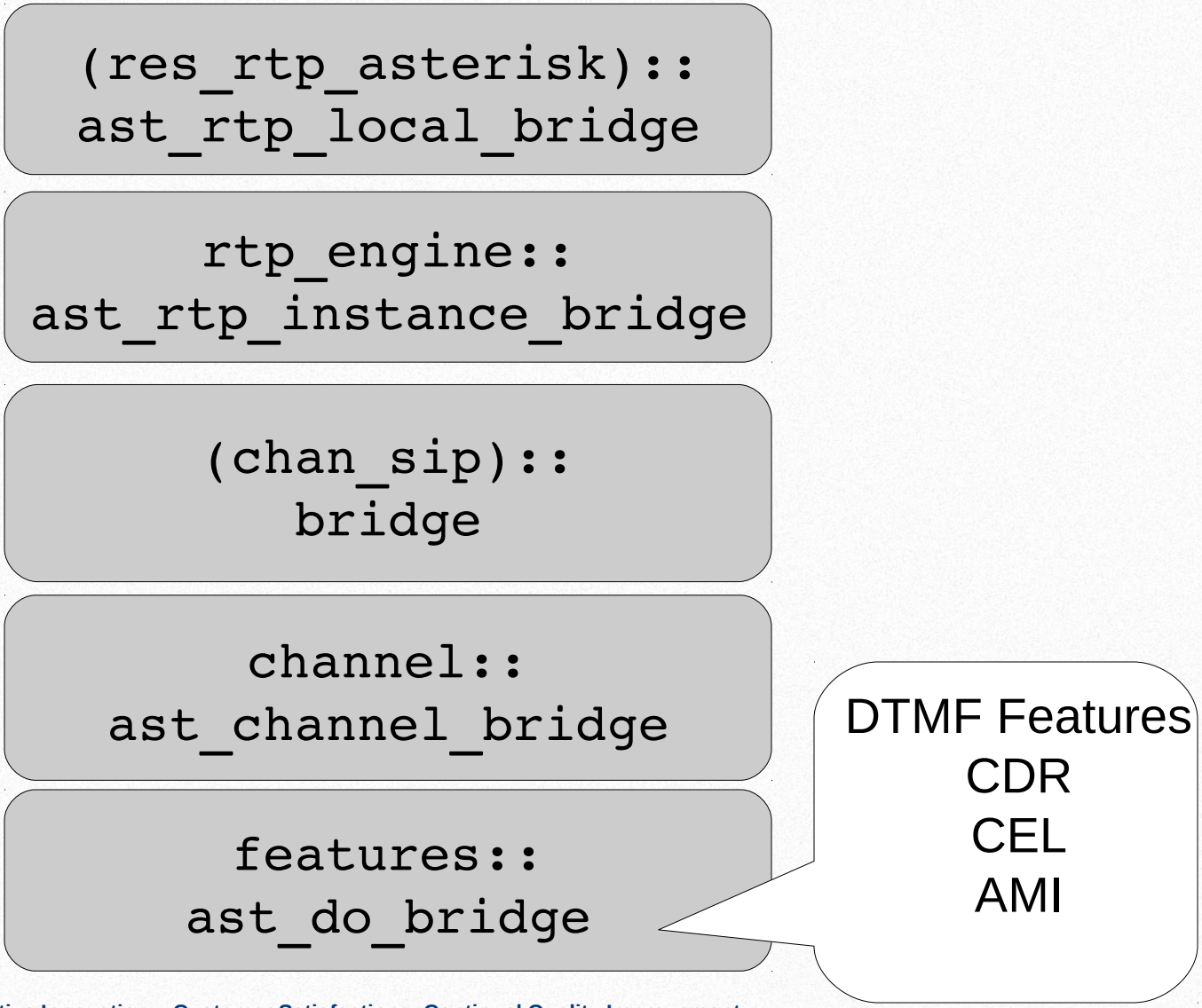
```
rtp_engine::
ast_rtp_instance_bridge
```

```
(chan_sip)::
bridge
```

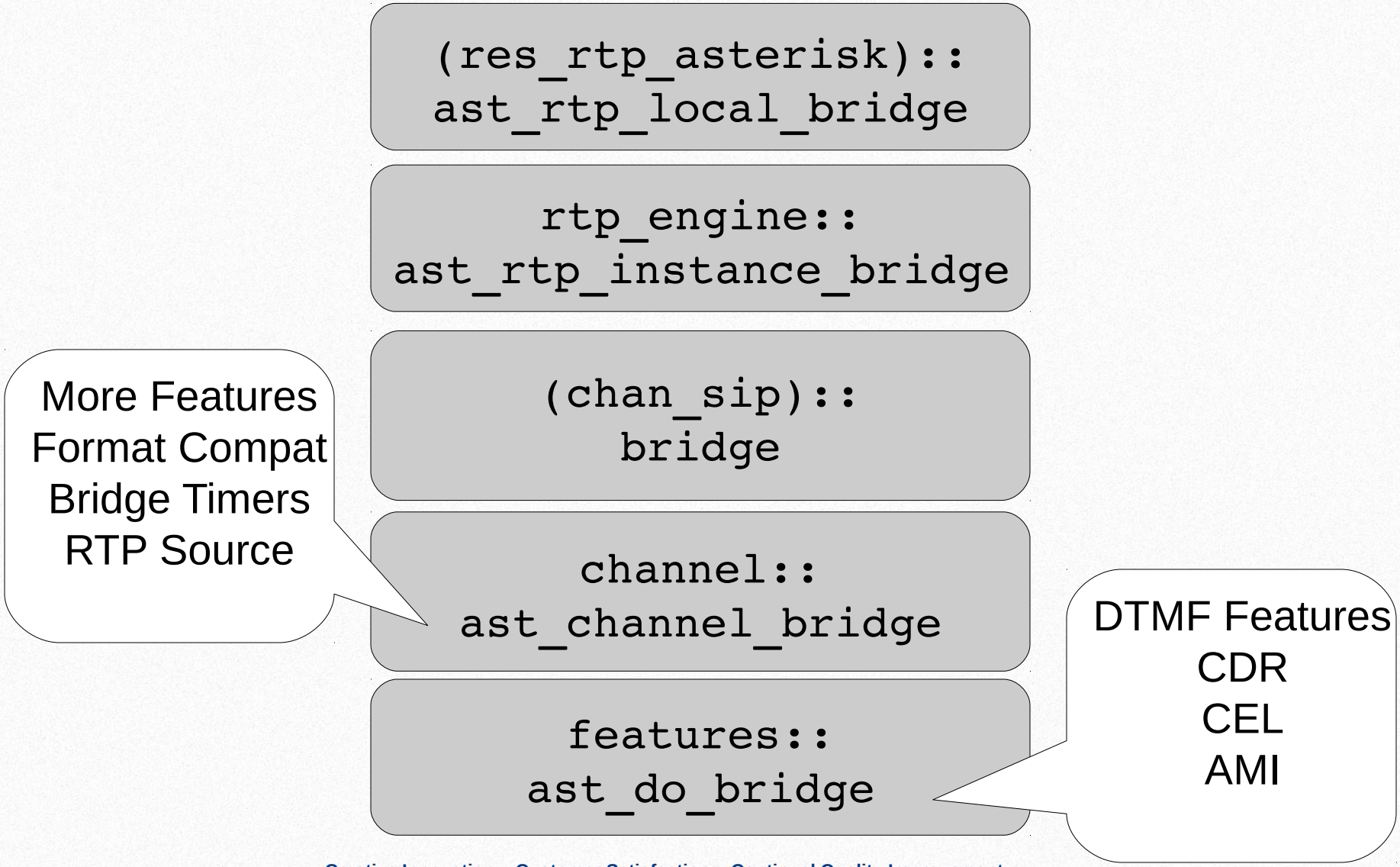
```
channel::
ast_channel_bridge
```

```
features::
ast_do_bridge
```

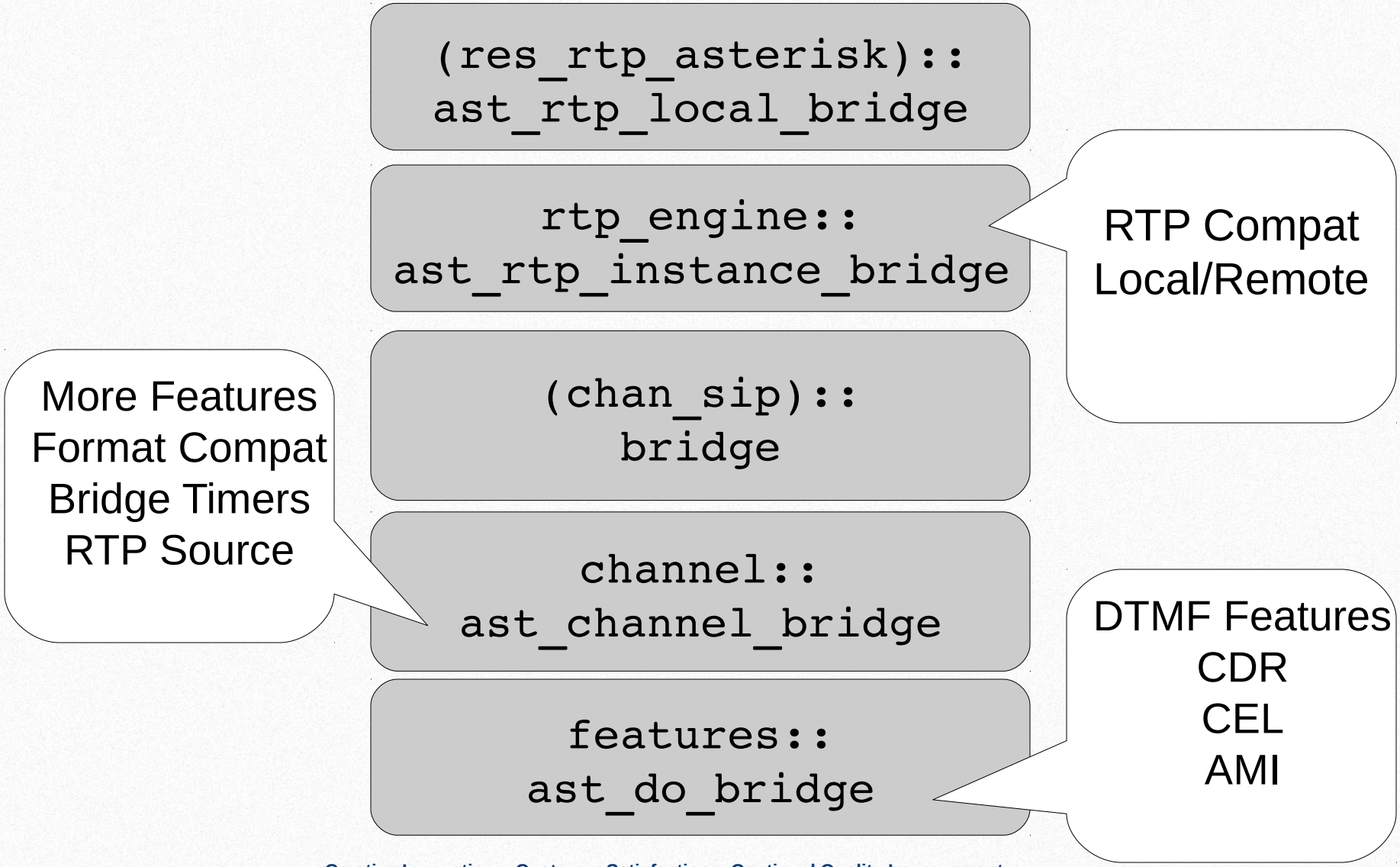
Problem: “Bridging”



Problem: “Bridging”



Problem: “Bridging”




THAT'S OKAY, THINGS ARE GOING TO BE OKAY.

This Is Not Fine

Constraints

- Channels must be “stable”
  - No rename events
  - No masquerade events
  - Predictable lifetime

Constraints

- Channels must be “stable”
  - No rename events
  - No masquerade events
  - Predictable lifetime
- Can't fundamentally change Asterisk's threading model
  - Asterisk is multithreaded, not multiprocess
  - The PBX Thread must still service an `ast_channel`

Constraints

- Channels must be “stable”
  - No rename events
  - No masquerade events
  - Predictable lifetime
- Can't fundamentally change Asterisk's threading model
  - Asterisk is multithreaded, not multiprocess
  - The PBX Thread must still service an `ast_channel`
- Use existing APIs to maximize chance of success
  - Bridging Framework (1.6.x)
  - Find places to *stop* expanding the scope

Version I: Bridging as an Object

Or: Let's try a novel approach and use some code we wrote five years ago

The Bridging Framework

- Bridges are an object
  - Maintain the state of the call
  - *Have* a mixing technology (strategy pattern)

The Bridging Framework

- Bridges are an object
  - Maintain the state of the call
  - *Have* a mixing technology (strategy pattern)
- Mixing strategies
  - “Simple” - two party, media in the core
  - “Softmix” - multiparty
  - “Multiplex” - multiple two party bridges

The Bridging Framework

- Bridges are an object
  - Maintain the state of the call
  - *Have* a mixing technology (strategy pattern)
- Mixing strategies
  - “Simple” - two party, media in the core
  - “Softmix” - multiparty
  - ~~“Multiplex” - multiple two party bridges~~

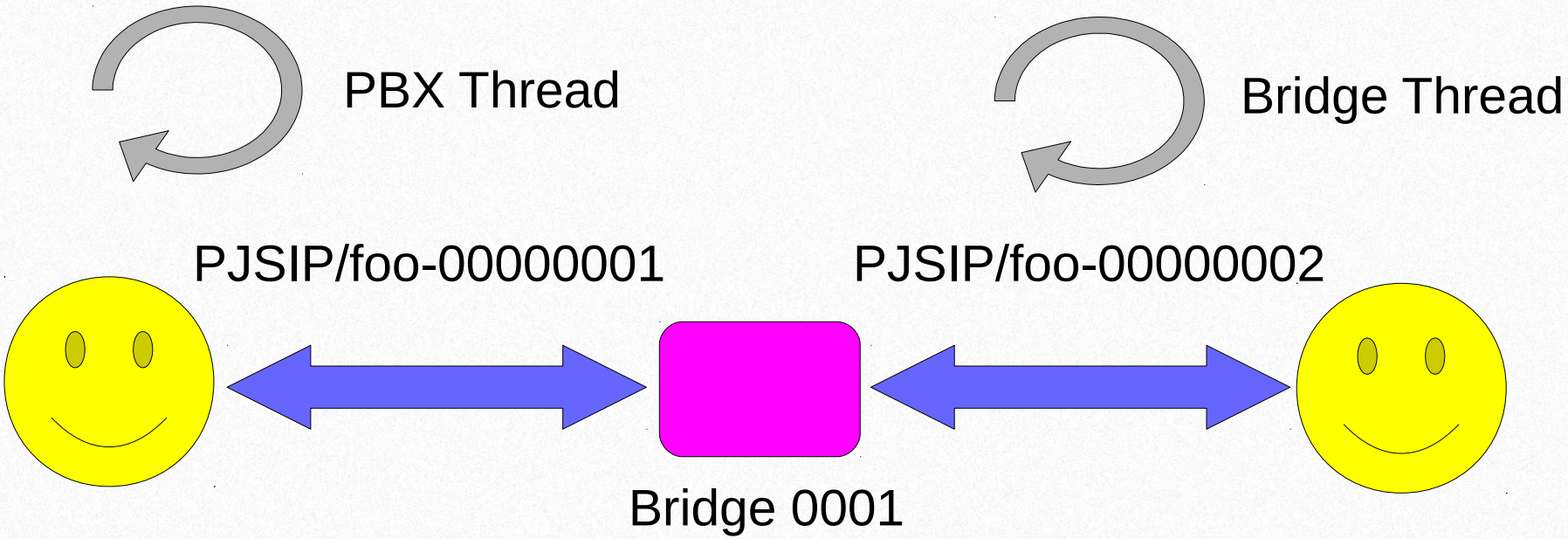
The Bridging Framework

- Bridges are an object
  - Maintain the state of the call
  - *Have* a mixing technology (strategy pattern)
- Mixing strategies
  - “Simple” - two party, media in the core
  - “Softmix” - multiparty
  - ~~“Multiplex” - multiple two party bridges~~
- Dedicated thread per channel
  - Allows channels to gracefully leave a bridge

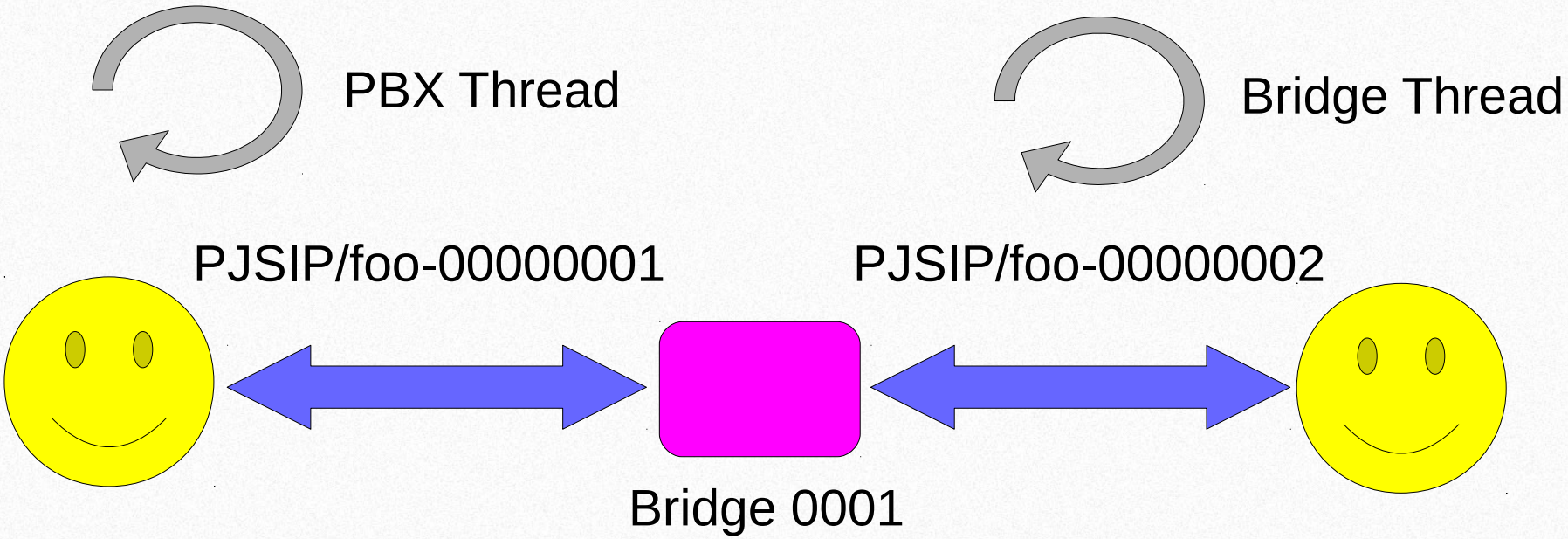
The Bridging Framework

- `ast_bridge_create` – make a new bridge
- `ast_bridge_join` – add a channel to a bridge (blocking)
- `ast_bridge_impart` – add a channel to a bridge (non-blocking)
- `ast_bridge_merge` – merge two bridges together
- `ast_bridge_move` – move a channel from one bridge to another
- `ast_bridge_transfer_blind` – blind transfer a channel to the dialplan
- `ast_bridge_transfer_attended` – perform an attended transfer between two channels

Releasing a Channel

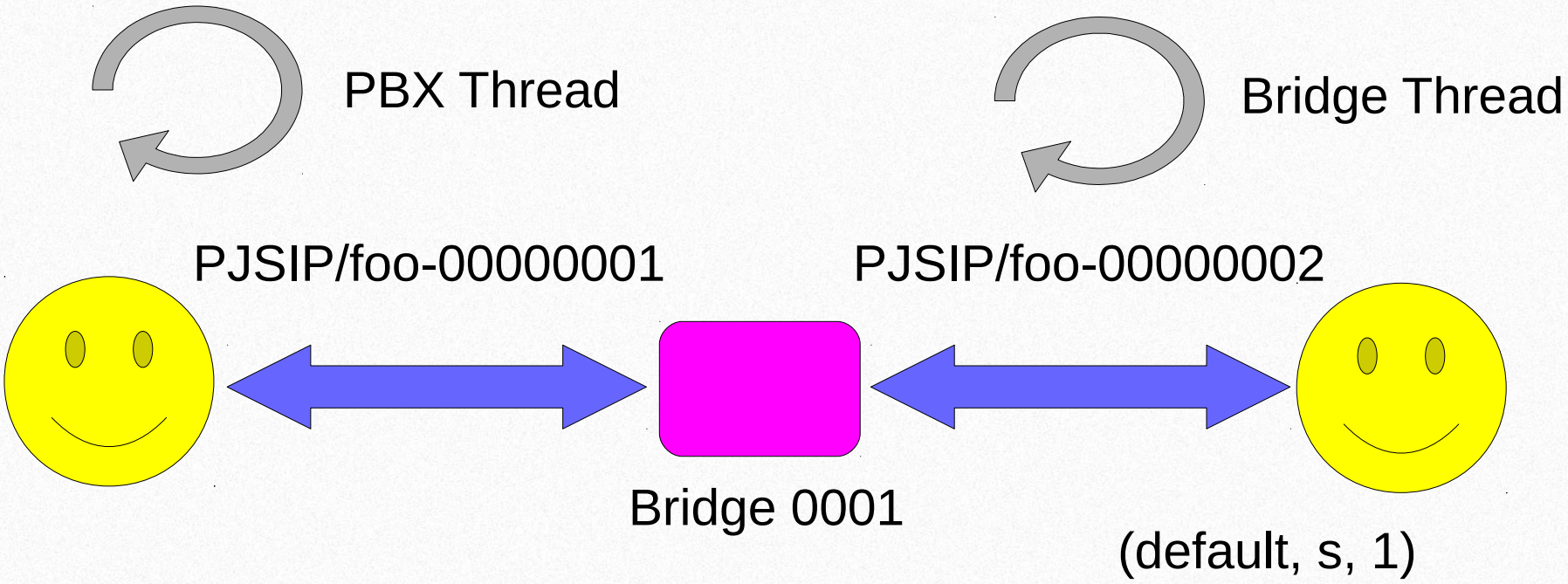


Releasing a Channel



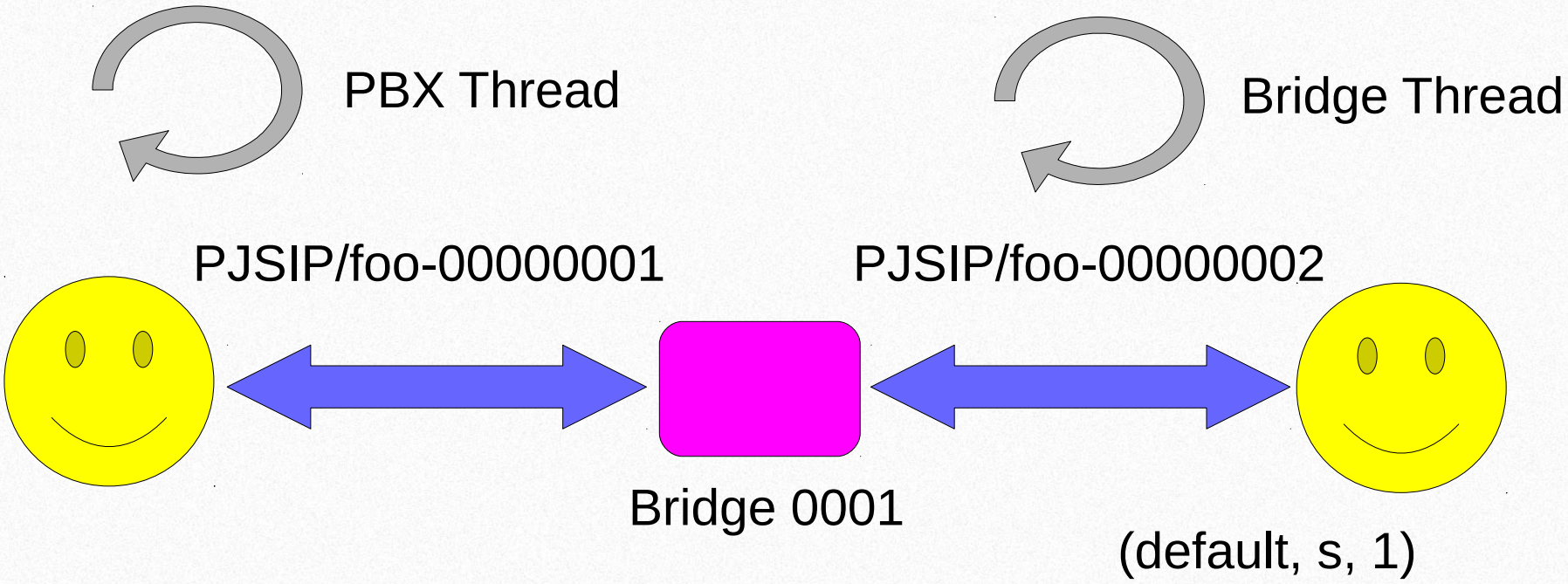

```
ast_bridge_set_after_goto(pjsip_foo,
    “default”, “s”, 1);
```

Releasing a Channel



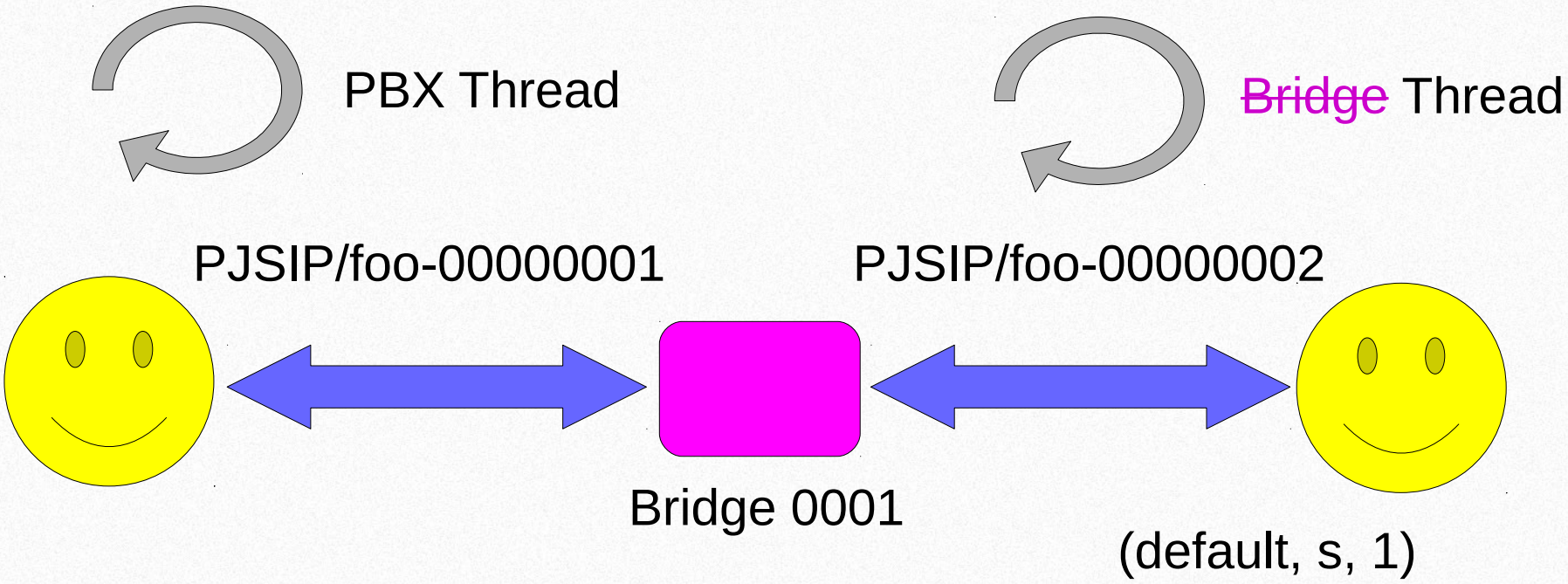

```
ast_bridge_set_after_goto(pjsip_foo,
    “default”, “s”, 1);
```

Releasing a Channel



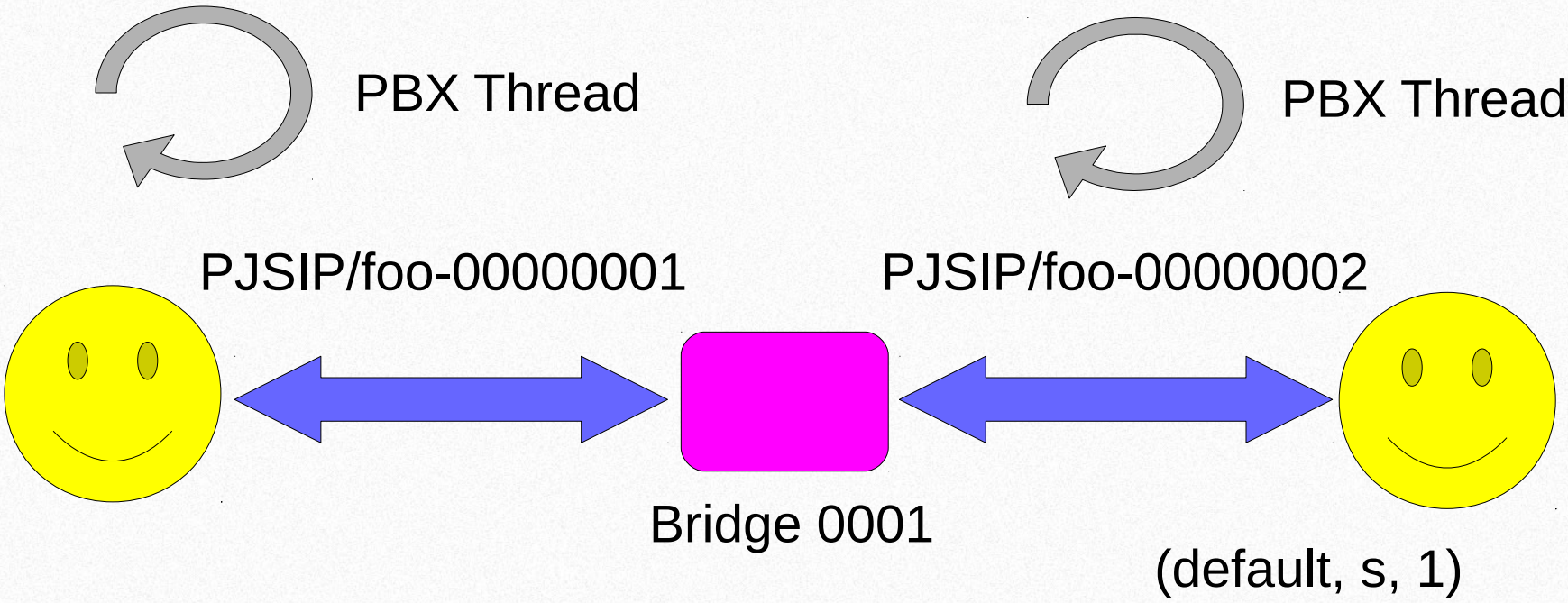

```
ast_bridge_kick(b01, pjsip_foo);
```

Releasing a Channel



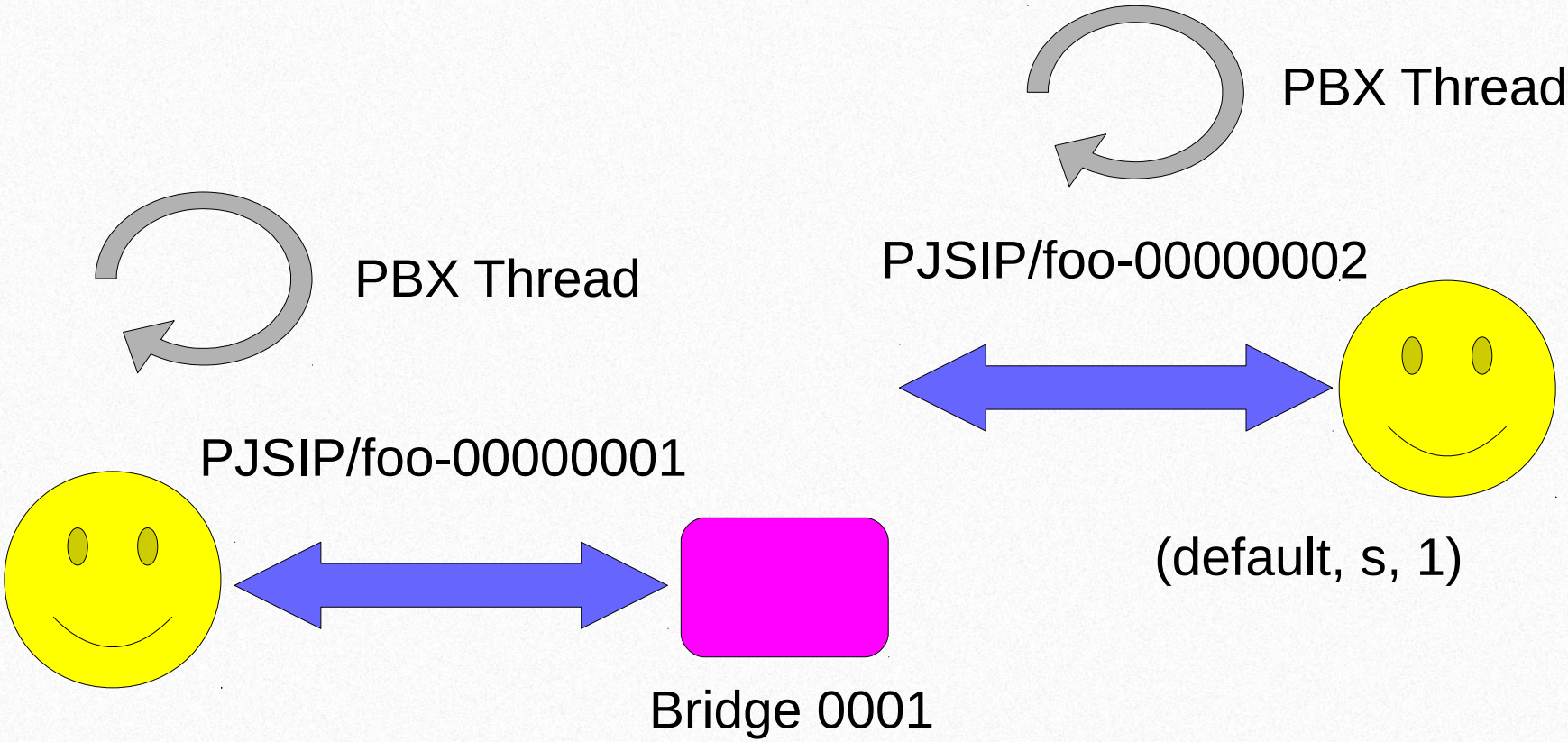

```
ast_bridge_kick(b01, pjsip_foo);
```

Releasing a Channel




```
ast_bridge_kick(b01, pjsip_foo);
```

Releasing a Channel




```
ast_bridge_kick(b01, pjsip_foo);
```

Bridging Events

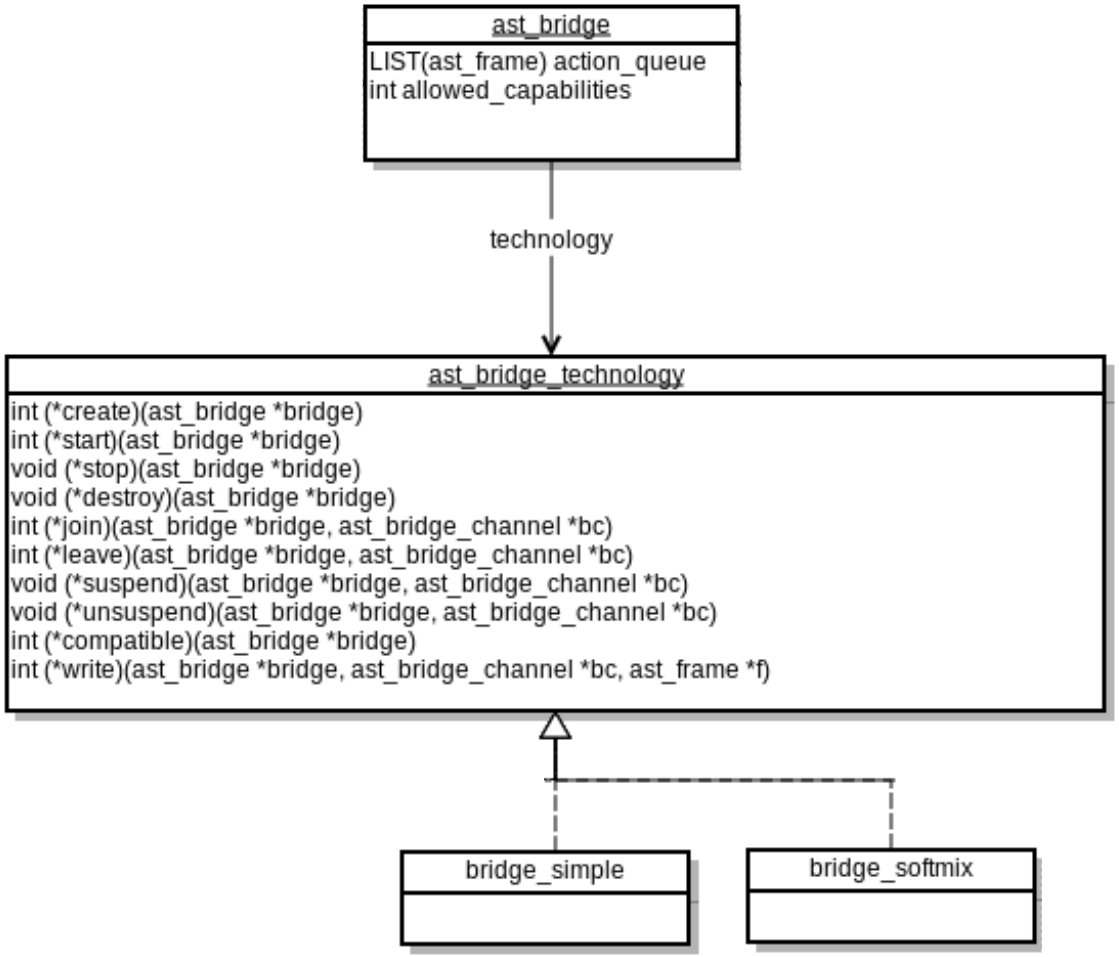
```
Event: BridgeEnter
Channel: PJSIP/foo-00000001
Bridge: 0001

Event: BridgeEnter
Channel: PJSIP/bar-00000002
Bridge: 0001

Event: BridgeLeave
Channel: PJSIP/bar-00000002
Bridge: 0001

Event: BridgeLeave
Channel: PJSIP/foo-00000001
Bridge: 0001
```

Version I


ast_bridge
LIST(ast_frame) action_queue
int allowed_capabilities
technology
ast_bridge_technology
int (*create)(ast_bridge *bridge)
int (*start)(ast_bridge *bridge)
void (*stop)(ast_bridge *bridge)
void (*destroy)(ast_bridge *bridge)
int (*join)(ast_bridge *bridge, ast_bridge_channel *bc)
int (*leave)(ast_bridge *bridge, ast_bridge_channel *bc)
void (*suspend)(ast_bridge *bridge, ast_bridge_channel *bc)
void (*unsuspend)(ast_bridge *bridge, ast_bridge_channel *bc)
int (*compatible)(ast_bridge *bridge)
int (*write)(ast_bridge *bridge, ast_bridge_channel *bc, ast_frame *f)
bridge_simple
bridge_softmix

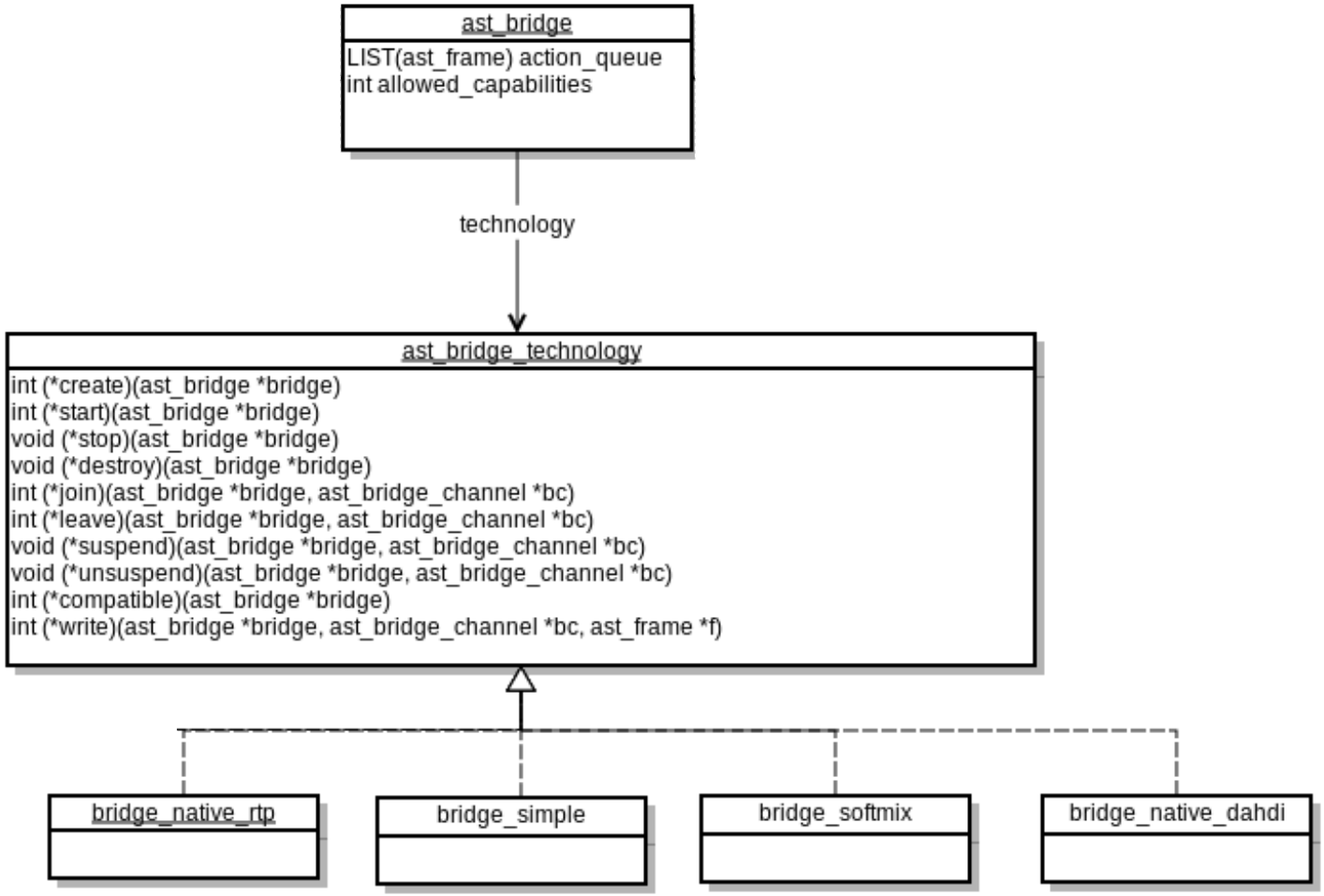
What about Native Bridging?

digium
The Asterisk Company

Version II: Native Bridging Modules

Or: Let's not rewrite res_rtp_asterisk just yet

Version II


ast_bridge
LIST(ast_frame) action_queue
int allowed_capabilities
technology
ast_bridge_technology
int (*create)(ast_bridge *bridge)
int (*start)(ast_bridge *bridge)
void (*stop)(ast_bridge *bridge)
void (*destroy)(ast_bridge *bridge)
int (*join)(ast_bridge *bridge, ast_bridge_channel *bc)
int (*leave)(ast_bridge *bridge, ast_bridge_channel *bc)
void (*suspend)(ast_bridge *bridge, ast_bridge_channel *bc)
void (*unsuspend)(ast_bridge *bridge, ast_bridge_channel *bc)
int (*compatible)(ast_bridge *bridge)
int (*write)(ast_bridge *bridge, ast_bridge_channel *bc, ast_frame *f)
bridge_native_rtp
bridge_simple
bridge_softmix
bridge_native_dahdi

Mixing Strategies

- PJSIP-foo joins
- Simple – 1 channel

Mixing Strategies

- PJSIP-foo joins
- DAHDI-i1 joins
- Simple – 1 channel
- Simple – 2 channels

Mixing Strategies

- PJSIP-foo joins
- DAHDI-i1 joins
- PJSIP-yack joins

- Simple – 1 channel
- Simple – 2 channels
- Softmix – 3 channels

Mixing Strategies

- PJSIP-foo joins
- DAHDI-i1 joins
- PJSIP-yack joins
- DAHDI-i1 leaves

- Simple – 1 channel
- Simple – 2 channels
- Softmix – 3 channels
- Native RTP (local) – 2 channels

Mixing Strategies

- PJSIP-foo joins
- DAHDI-i1 joins
- PJSIP-yack joins
- DAHDI-i1 leaves
- PJSIP-bar joins

- Simple – 1 channel
- Simple – 2 channels
- Softmix – 3 channels
- Native RTP (local) – 2 channels
- Softmix – 3 channels

Mixing Strategies

- PJSIP-foo joins
- DAHDI-i1 joins
- PJSIP-yack joins
- DAHDI-i1 leaves
- PJSIP-bar joins
- PJSIP-yack leaves

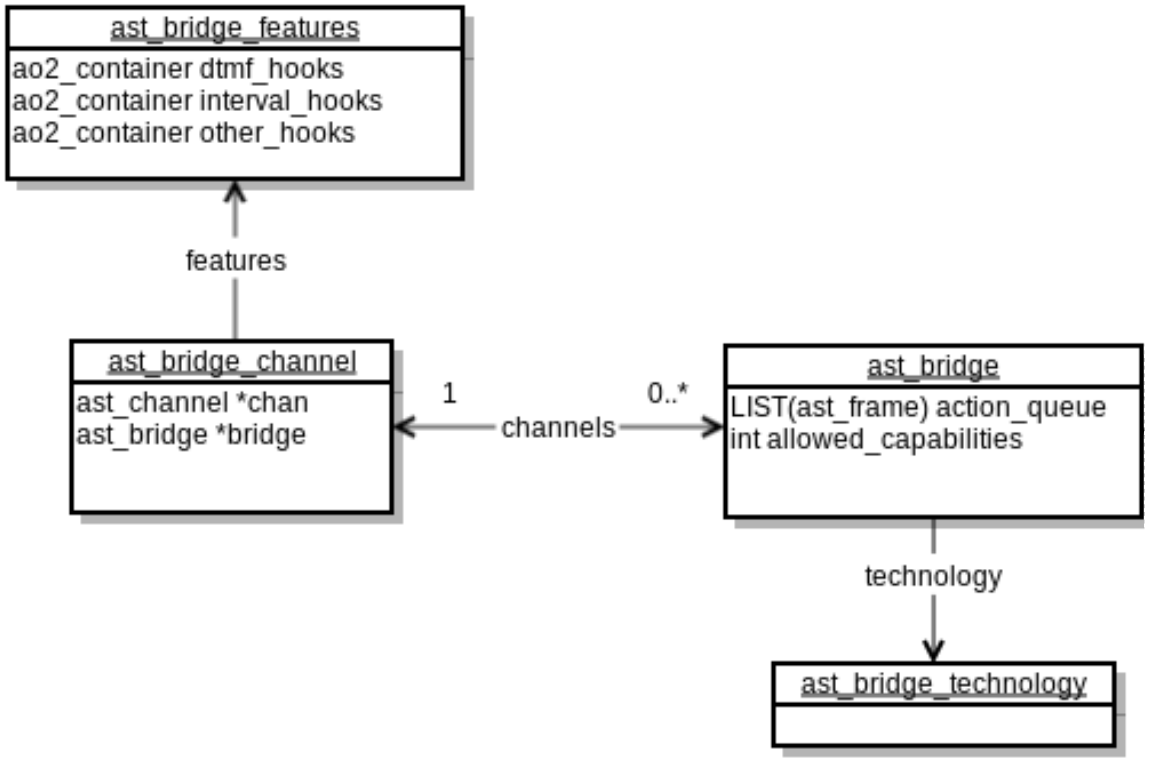
- Simple – 1 channel
- Simple – 2 channels
- Softmix – 3 channels
- Native RTP (local) – 2 channels
- Softmix – 3 channels
- Native RTP (rmt) – 2 channels

What about Features?

Version III: Features

Or: Users like buttons

Version III


ast_bridge_features
ao2_container dtmf_hooks
ao2_container interval_hooks
ao2_container other_hooks
features
ast_bridge_channel
ast_channel *chan
ast_bridge *bridge
1
channels
0..*
ast_bridge
LIST(ast_frame) action_queue
int allowed_capabilities
technology
ast_bridge_technology

What about Parking?

Version IV: Parking as a Bridge

Or: Hold All the Things

Parking

- Create a new bridge mixing technology, 'holding'
  - Drops all media read from channels
  - Entertains channels with music, ringing, or white noise

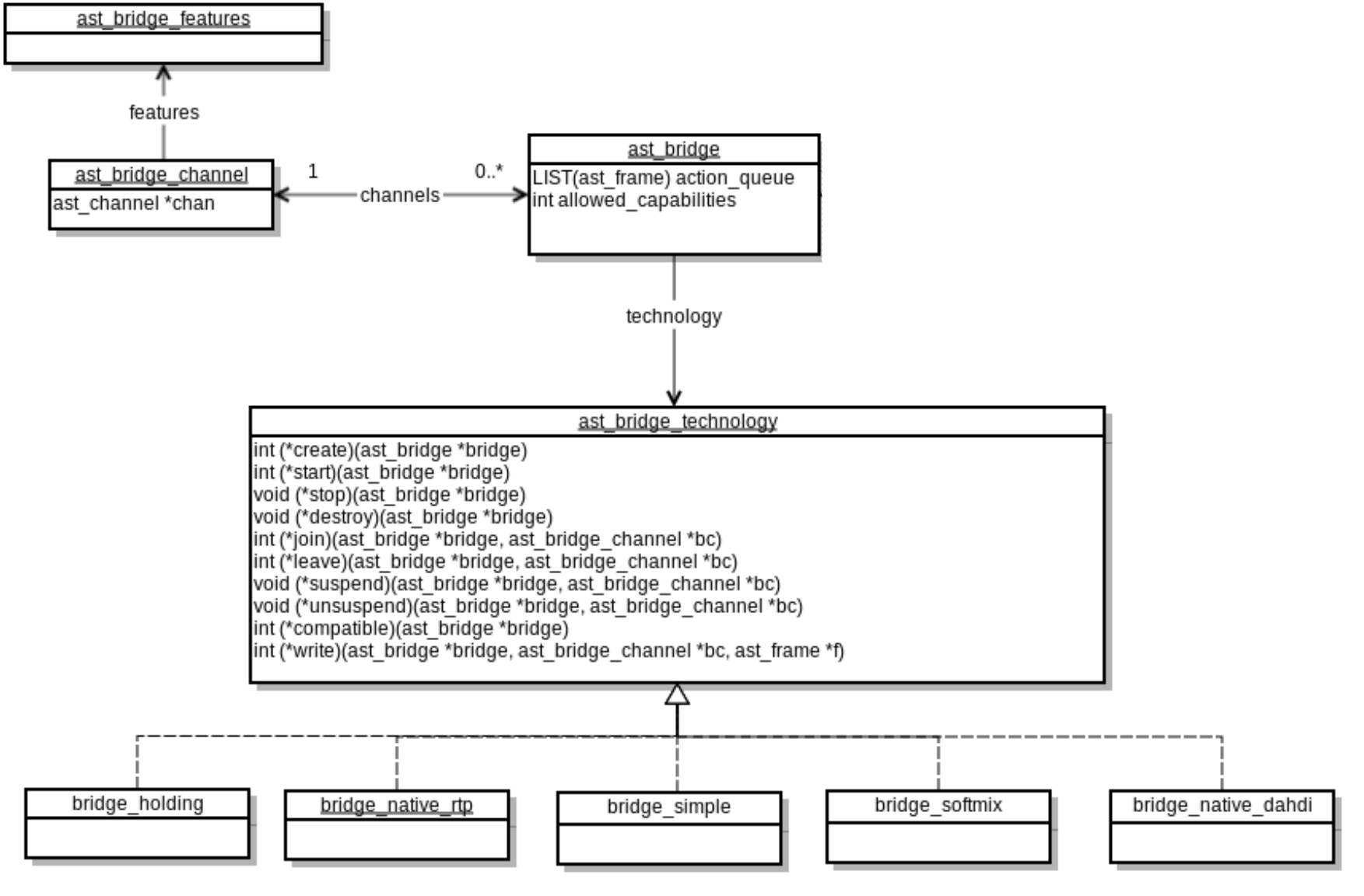
Parking

- Create a new bridge mixing technology, 'holding'
  - Drops all media read from channels
  - Entertains channels with music, ringing, or white noise
- Announcements
  - Channels join either as participant or announcer
  - Announcer channels broadcast their media to all participants

Parking

- Create a new bridge mixing technology, 'holding'
  - Drops all media read from channels
  - Entertains channels with music, ringing, or white noise
- Announcements
  - Channels join either as participant or announcer
  - Announcer channels broadcast their media to all participants
- Parking
  - Park puts channel into holding bridge
  - Pickup pulls channel from bridge and puts it into a standard 'mixing' bridge with the pickup-ee

Version IV


ast_bridge_features
features
ast_bridge_channel
ast_channel *chan
1
channels
0..*
ast_bridge
LIST(ast_frame) action_queue
int allowed_capabilities
technology
ast_bridge_technology
int (*create)(ast_bridge *bridge)
int (*start)(ast_bridge *bridge)
void (*stop)(ast_bridge *bridge)
void (*destroy)(ast_bridge *bridge)
int (*join)(ast_bridge *bridge, ast_bridge_channel *bc)
int (*leave)(ast_bridge *bridge, ast_bridge_channel *bc)
void (*suspend)(ast_bridge *bridge, ast_bridge_channel *bc)
void (*unsuspend)(ast_bridge *bridge, ast_bridge_channel *bc)
int (*compatible)(ast_bridge *bridge)
int (*write)(ast_bridge *bridge, ast_bridge_channel *bc, ast_frame *f)
bridge_holding
bridge_native_rtp
bridge_simple
bridge_softmix
bridge_native_dahdi

What about CDRs/CEL/AMI?

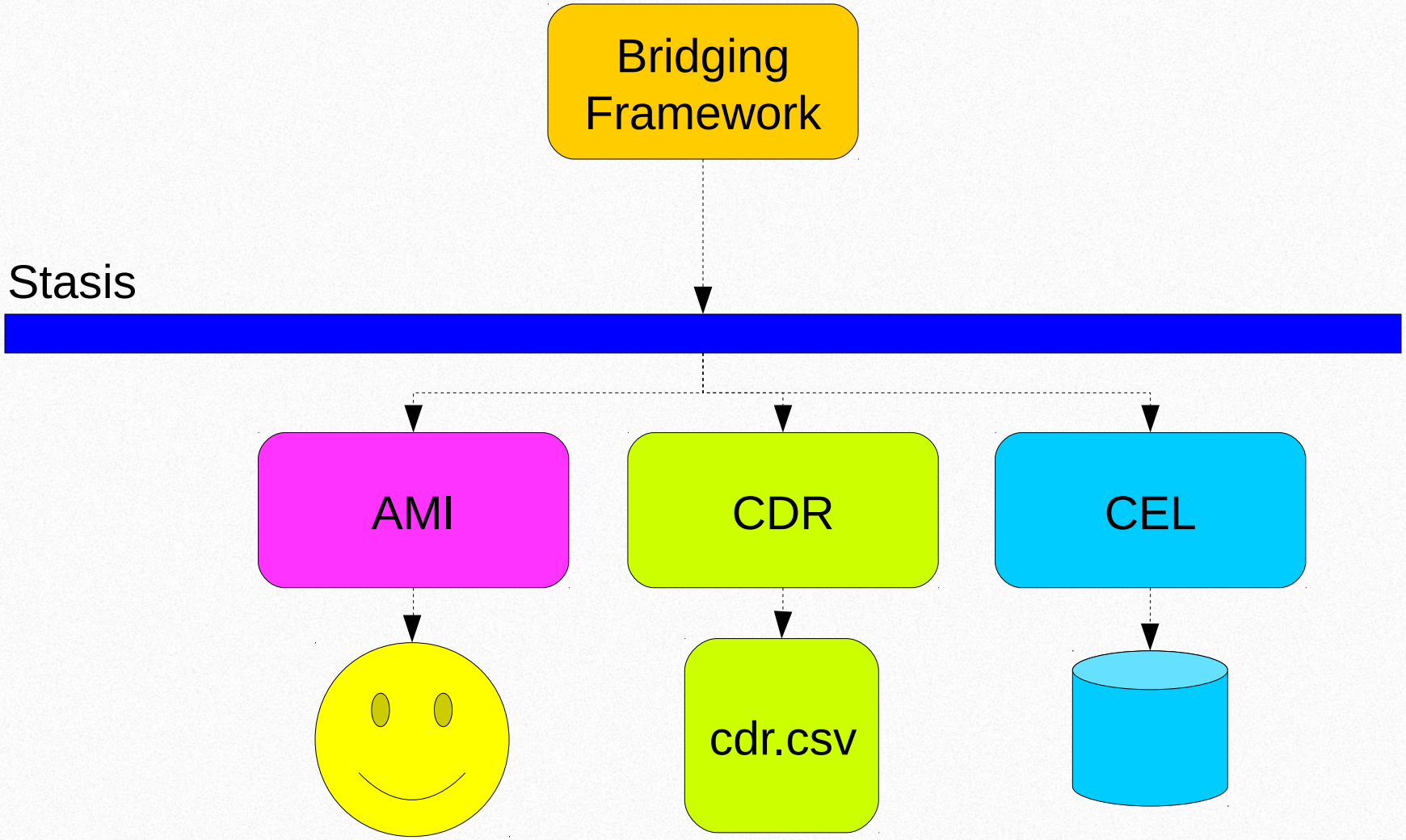
Version V: Stasis

Or: If we never charged people money this would be much easier

Consuming Bridge State

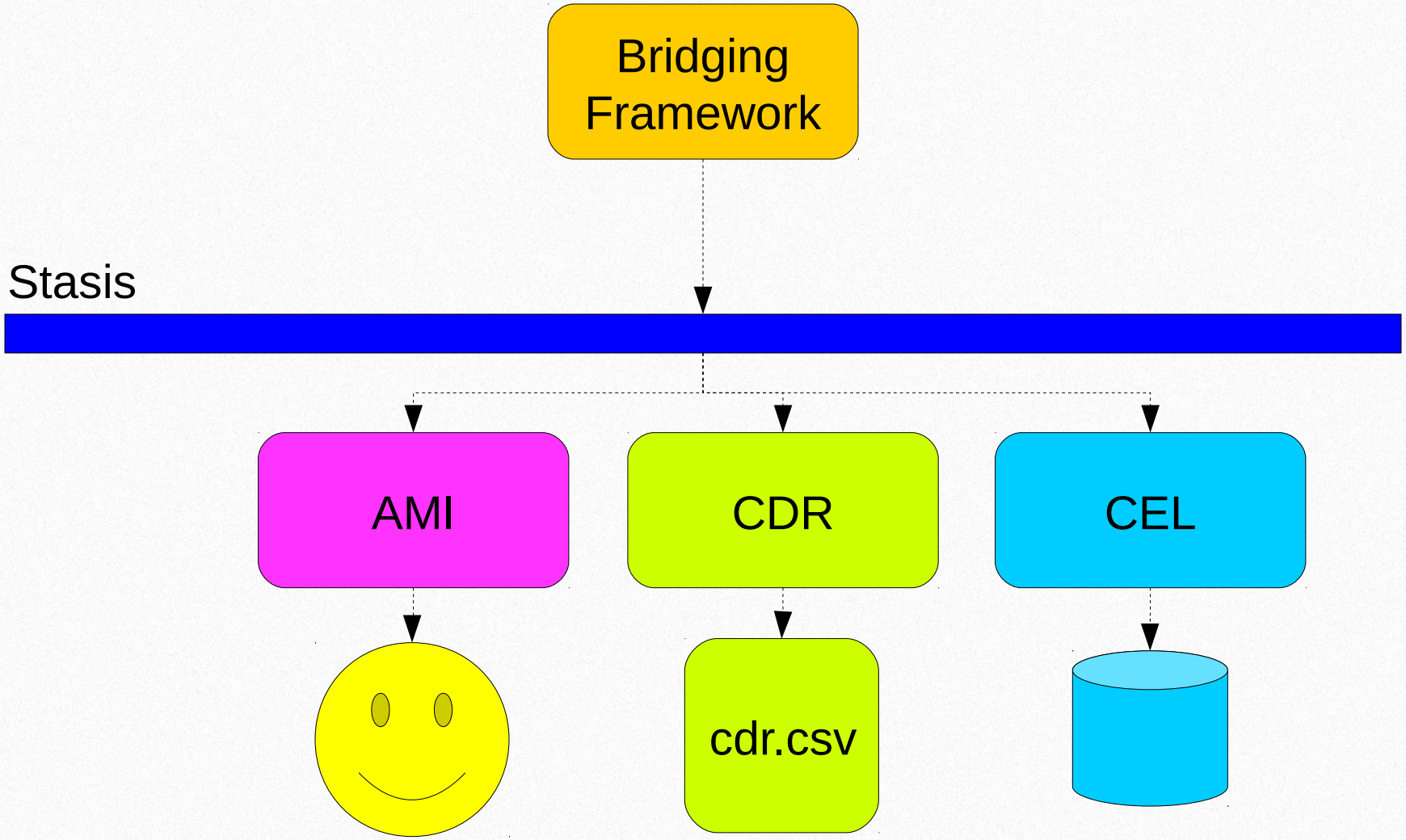
- Do *not* put CDR, CEL, AMI logic back into Bridging Framework
- Stasis: Internal pub/sub message bus
  - Bridge state is published onto bus
  - AMI, CDR, CEL become consumers of that state

Consuming Bridge State



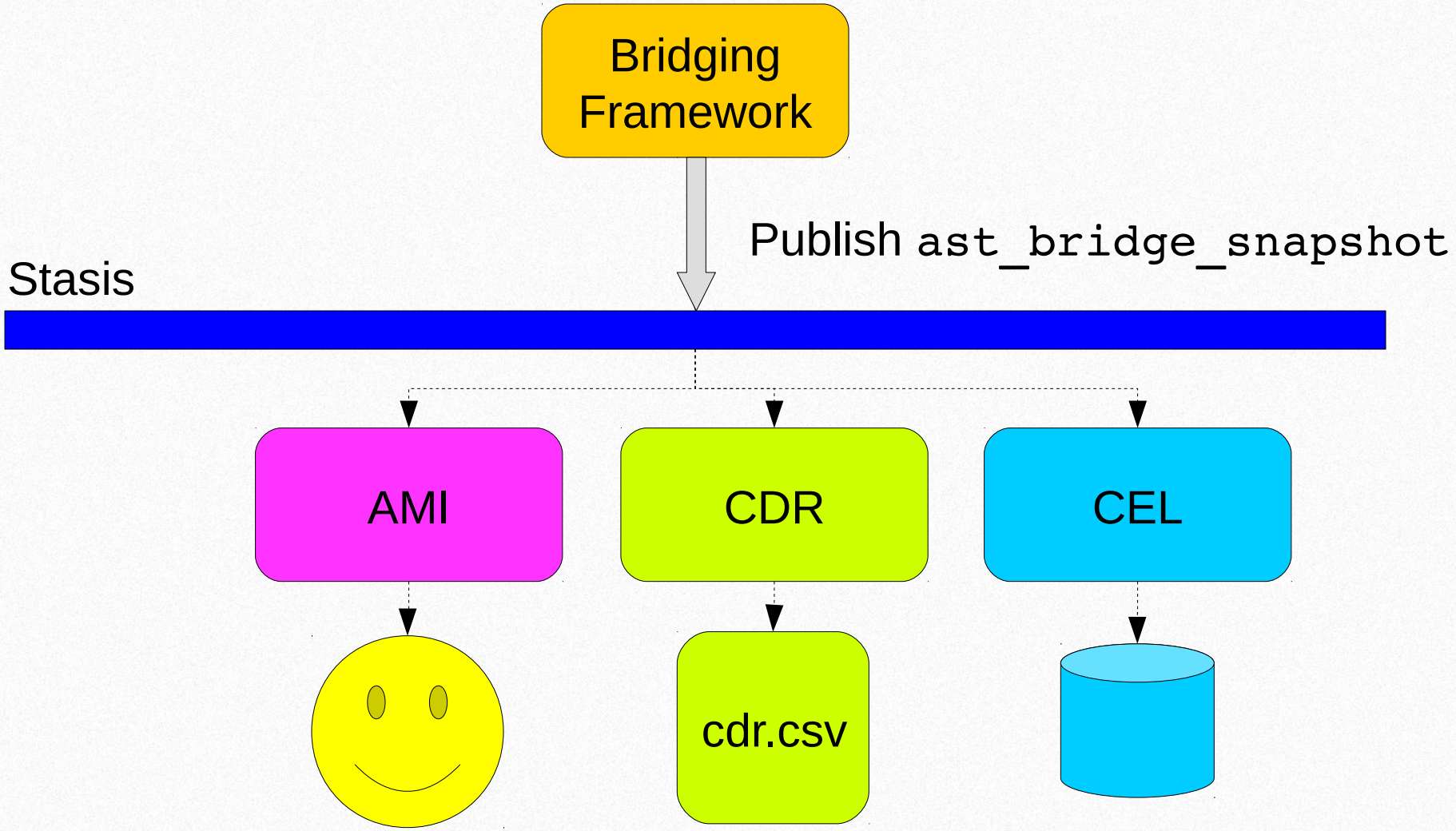
Consuming Bridge State

Channel pulled from bridge



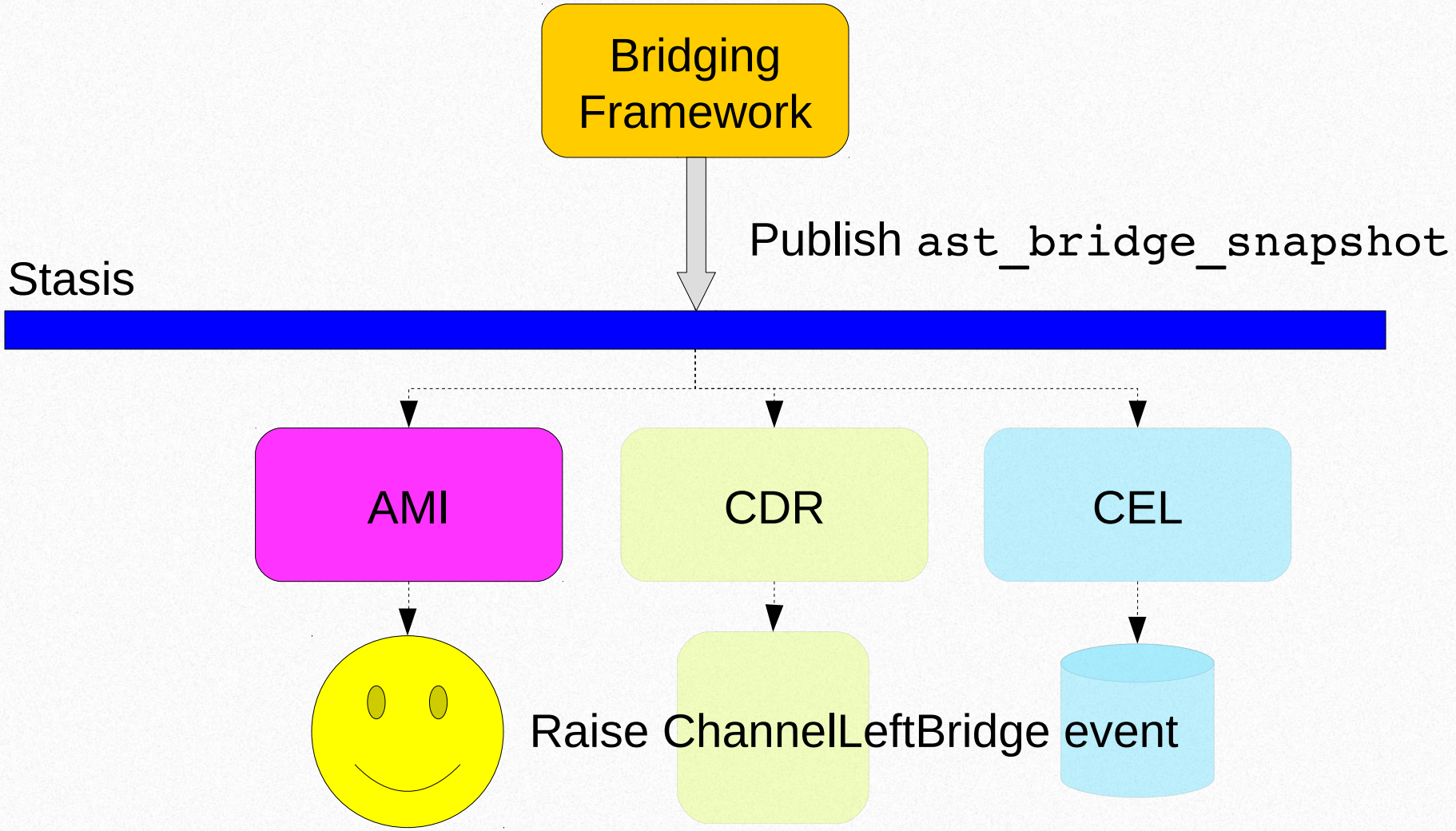
Consuming Bridge State

Channel pulled from bridge



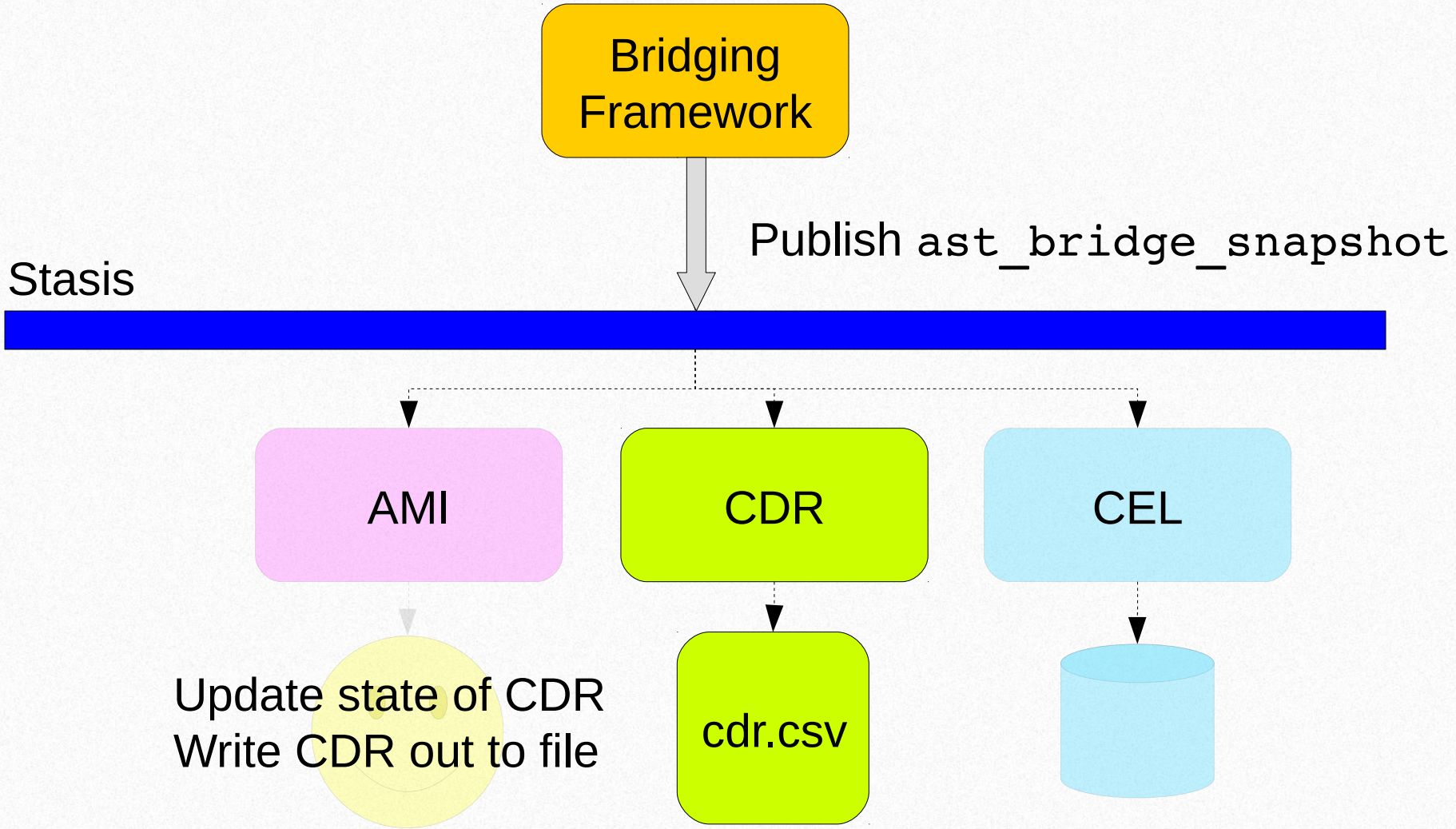
Consuming Bridge State

Channel pulled from bridge



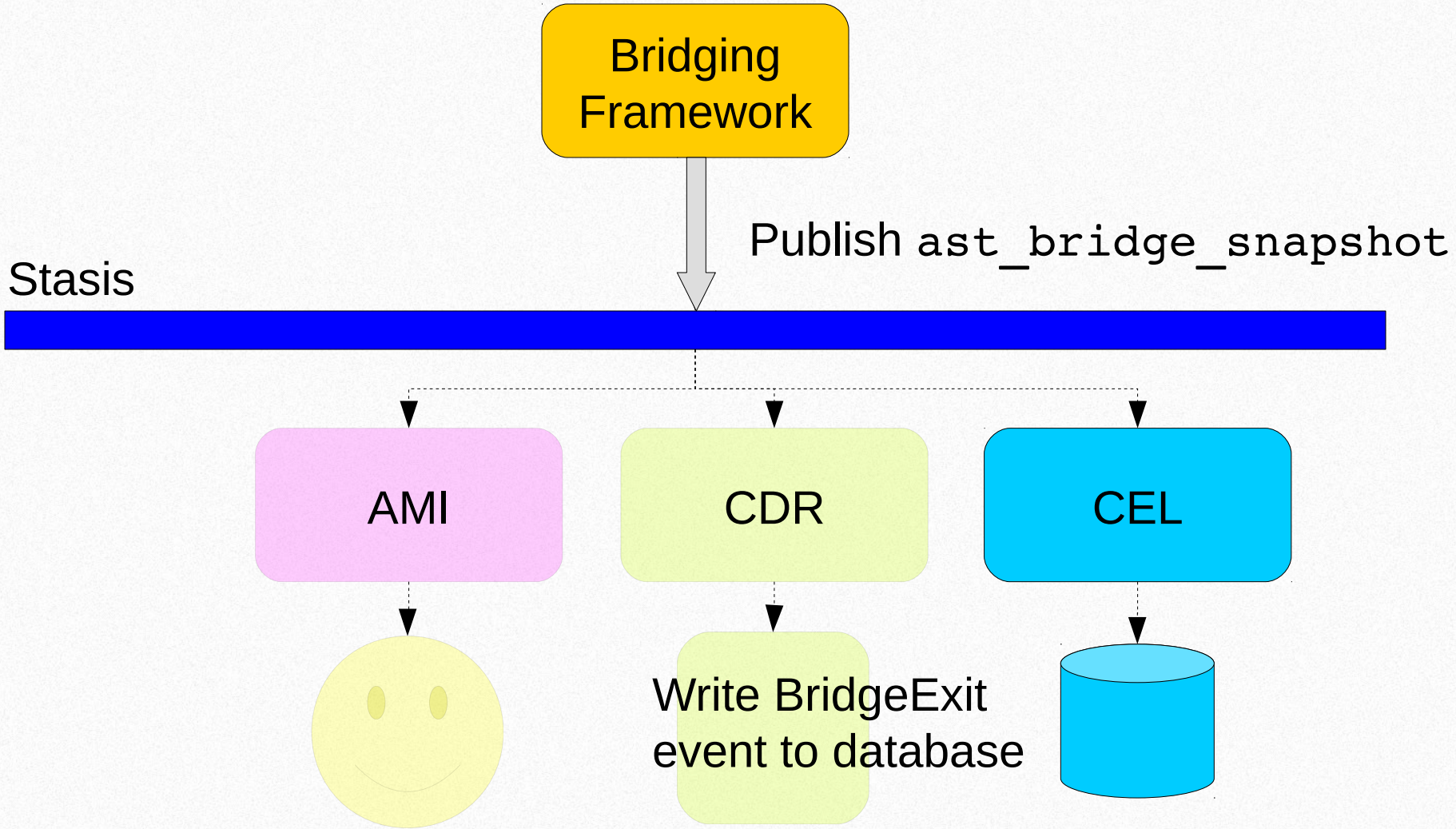
Consuming Bridge State

Channel pulled from bridge



Consuming Bridge State

Channel pulled from bridge

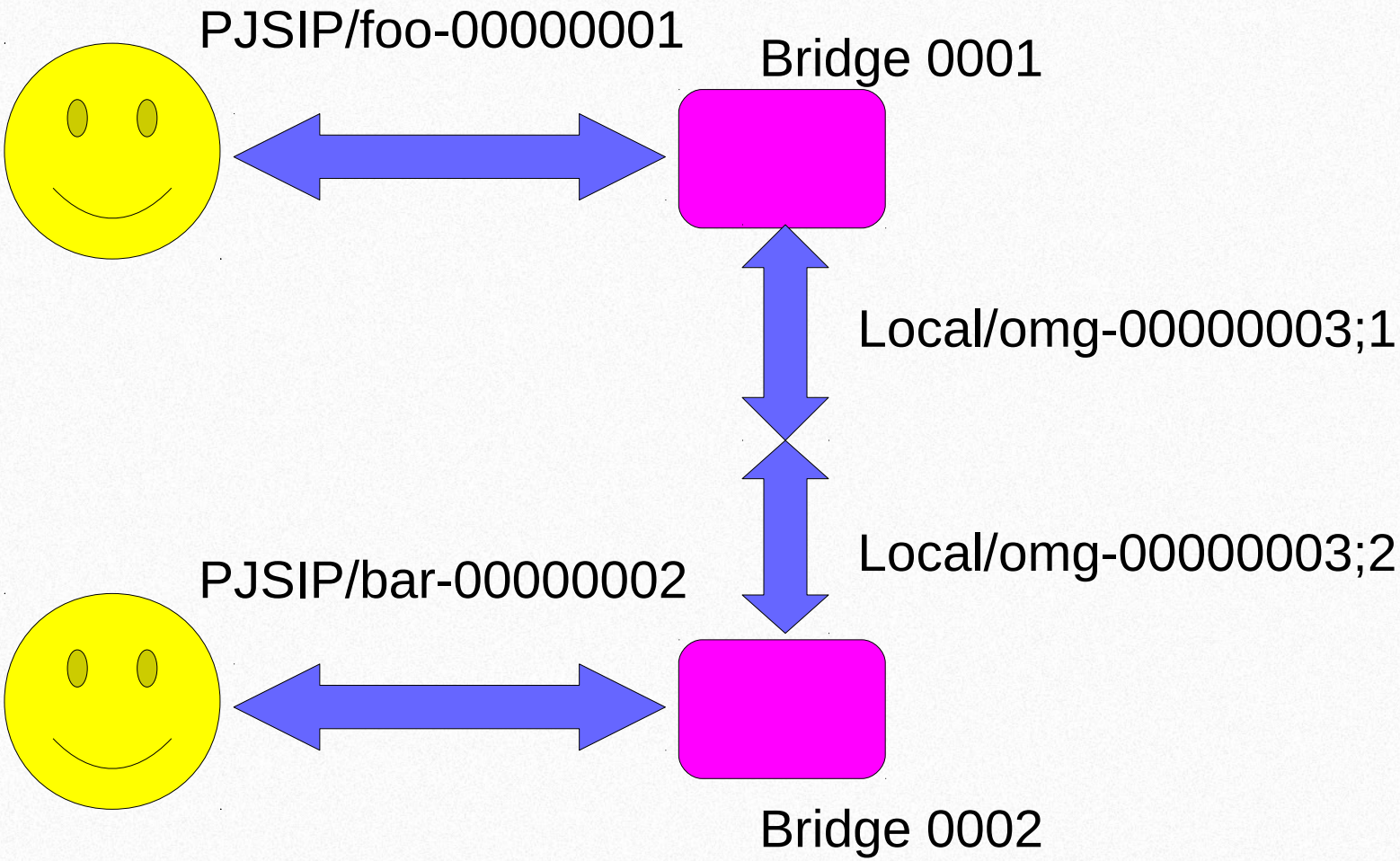


What about Local Channel Optimization?

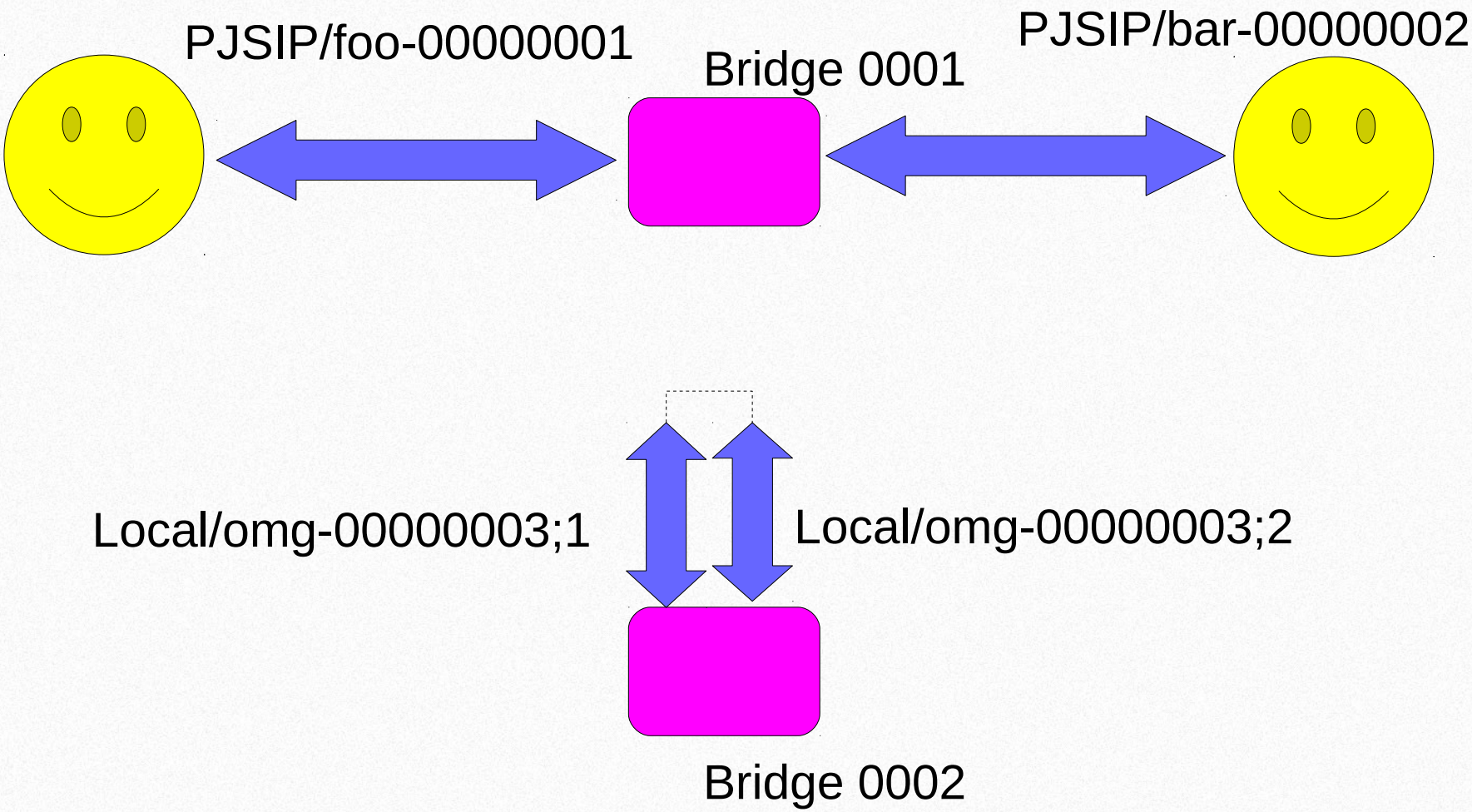
Version VI: Local Channels

Or: Oh Snap.

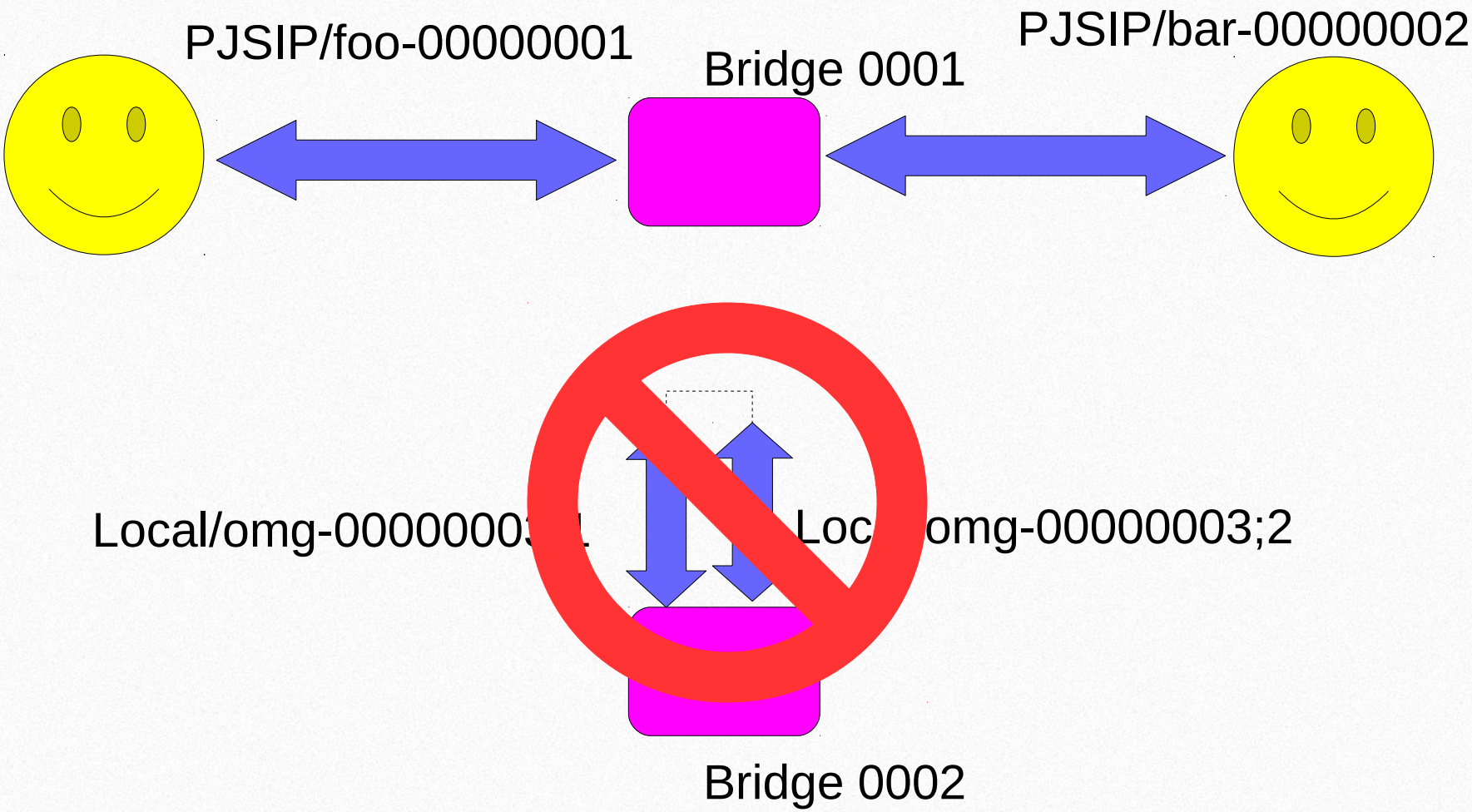
Optimization: Bridge Swap



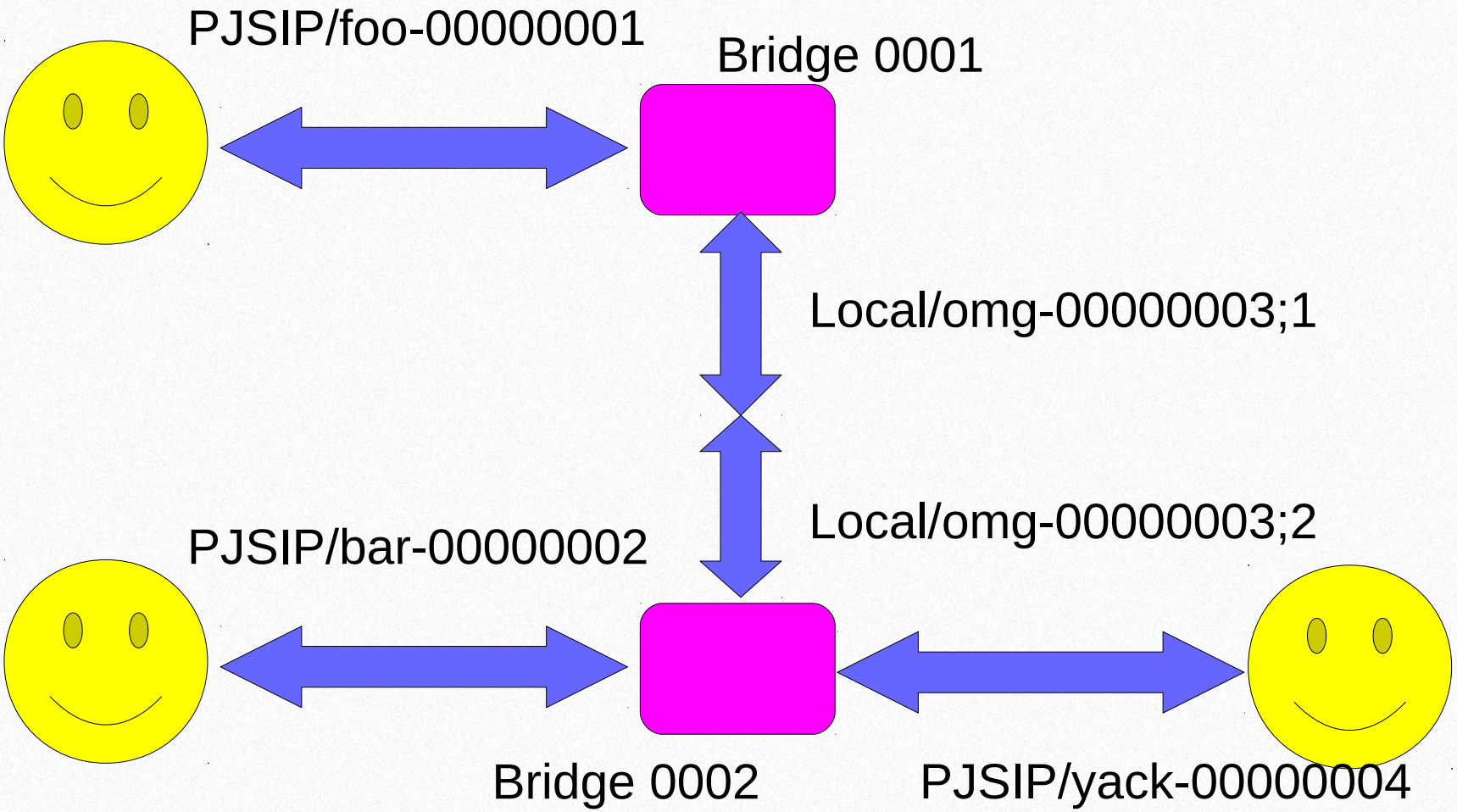
Optimization: Bridge Swap



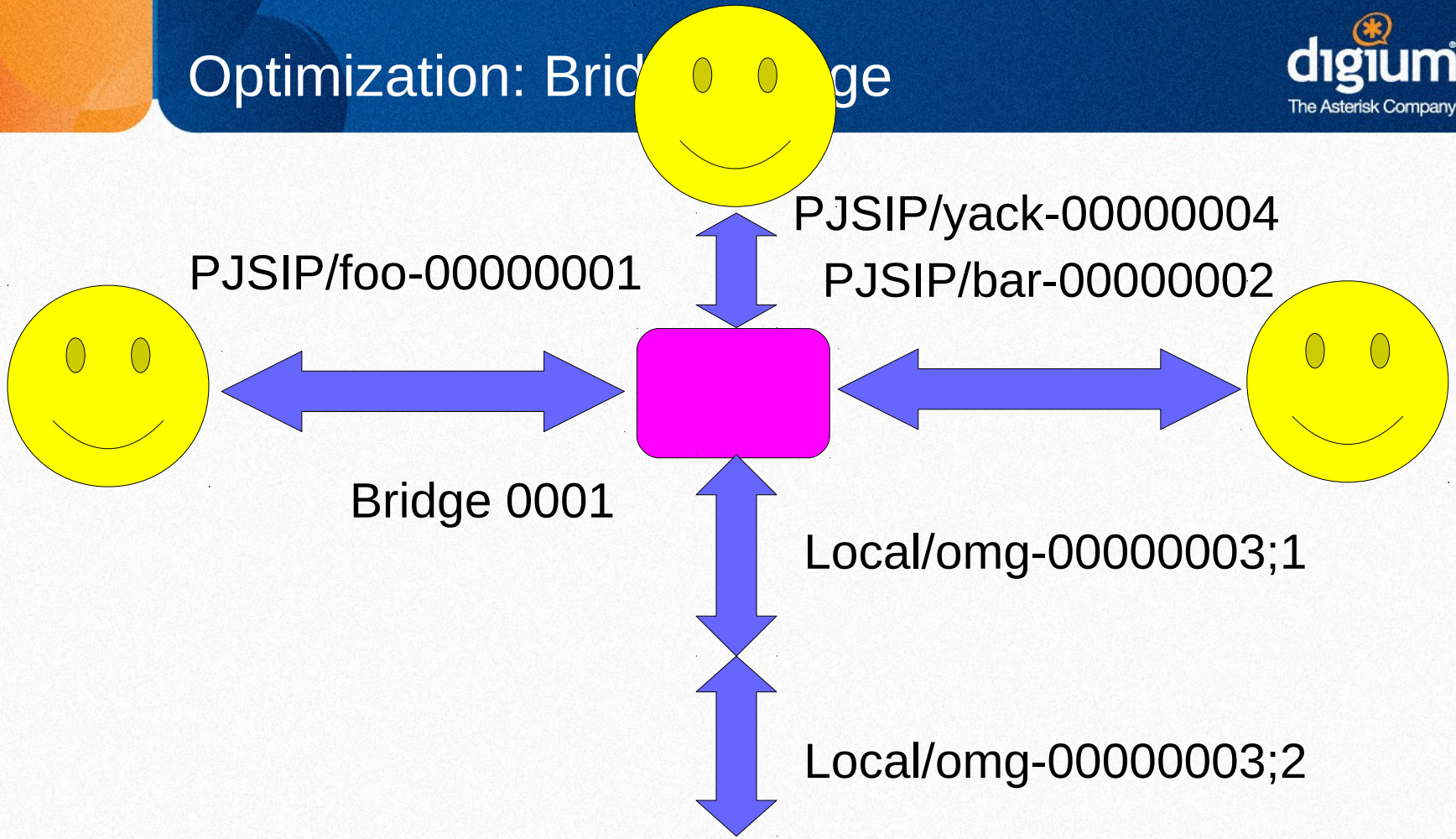
Optimization: Bridge Swap



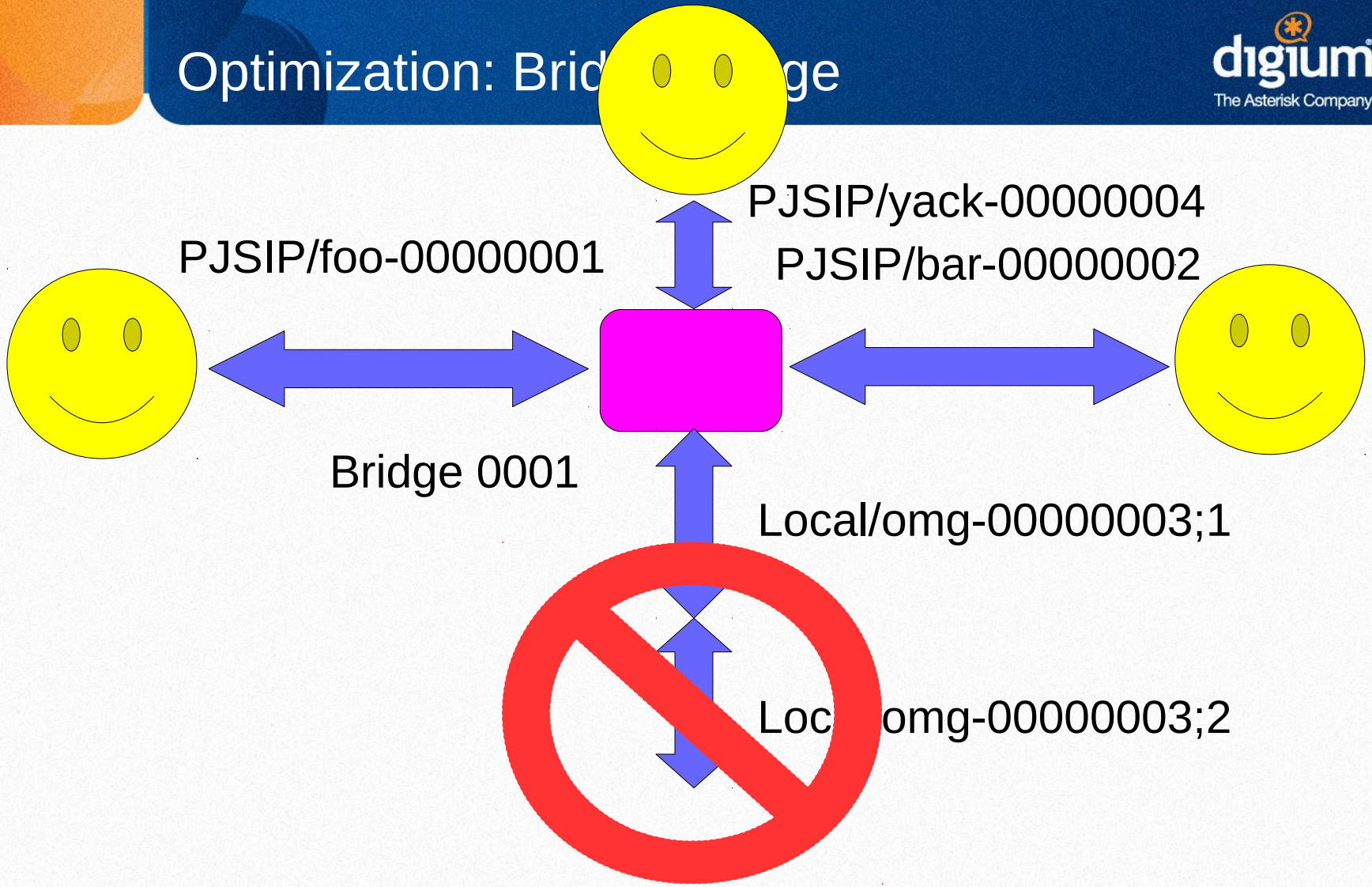
Optimization: Bridge Merge



Optimization: Brid ge



Optimization: Bridge



What happens when a Local channel merges into ConfBridge?

Version VII: Class Bridges

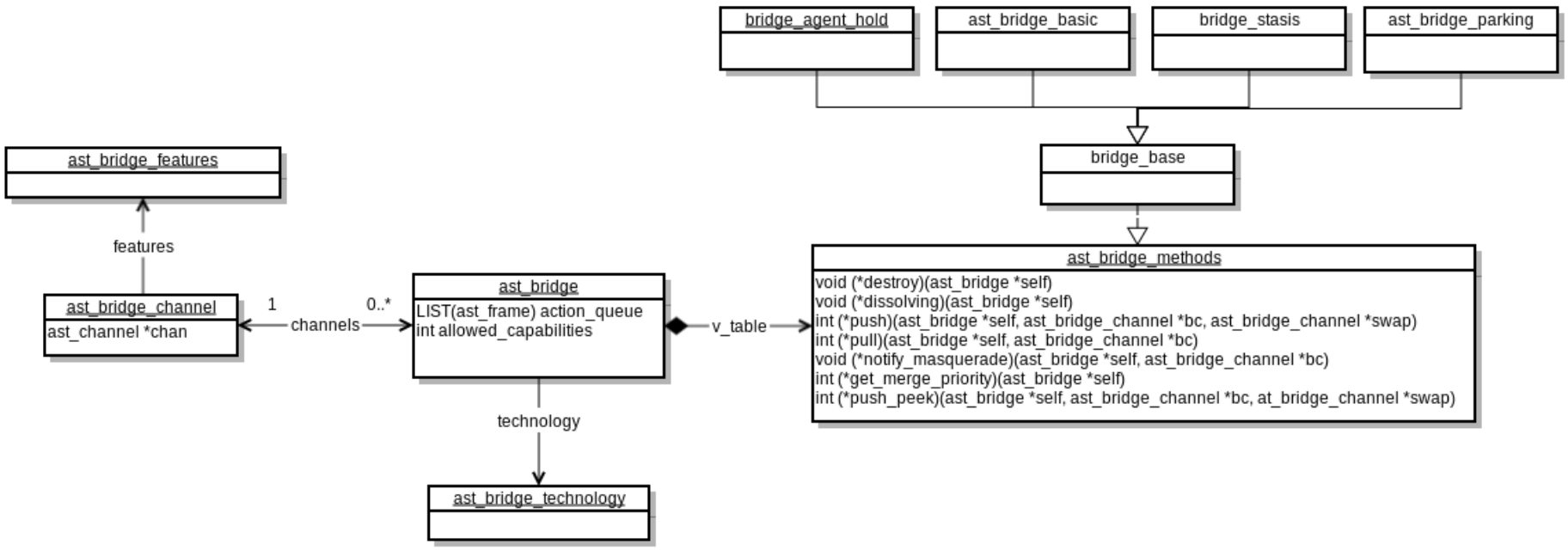
Or: That's it, I'm taking my ball and going home now.

Class Bridges

- Some bridges have application state in addition to their mixing strategy
  - ConfBridge: marked users, admins, etc.
  - Parking: which parking lot am I in
  - Stasis: ARI specific logic to know who controls a channel
  - Agent waiting pools: how long has a Queue agent been waiting for a call

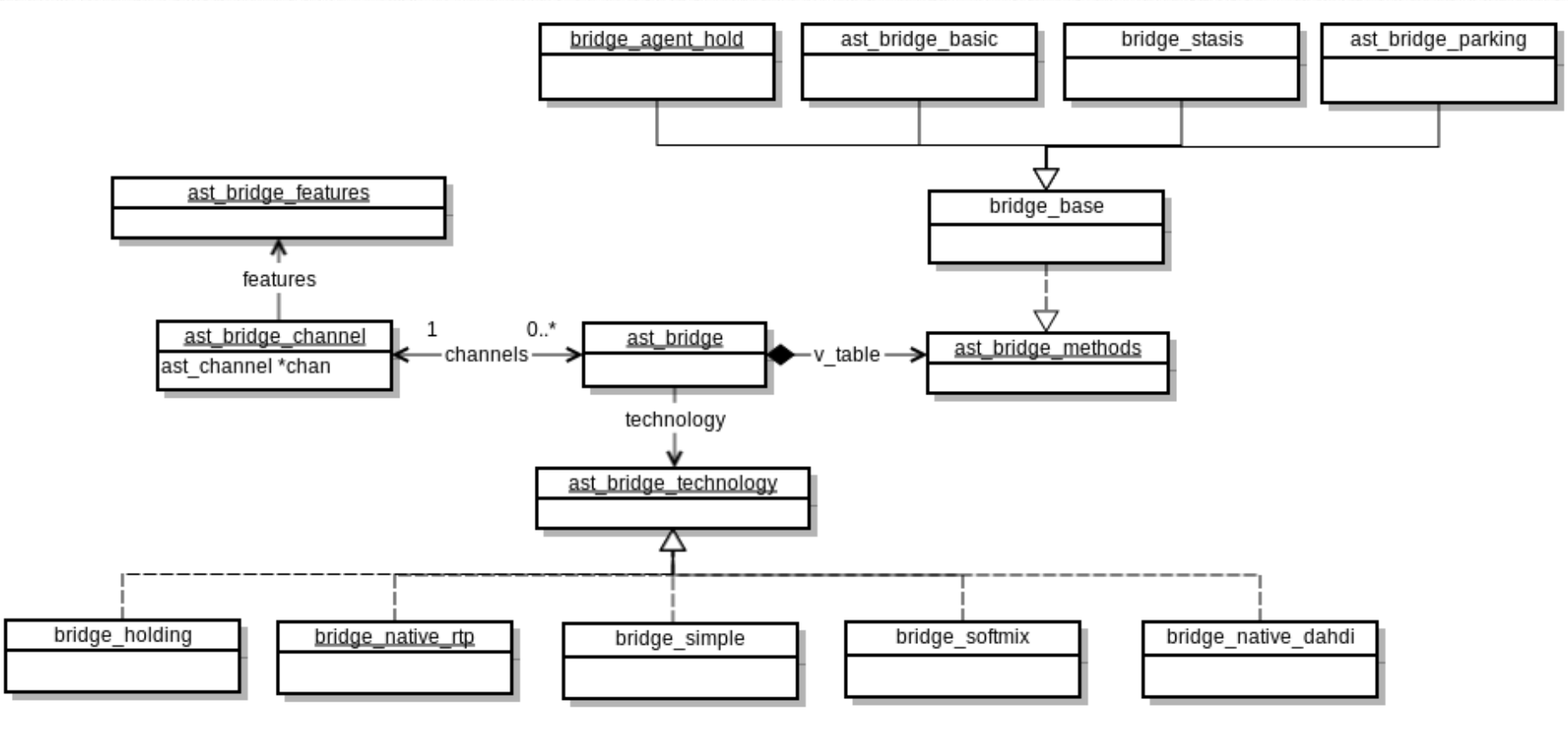
Class Bridges

- Some bridges have application state in addition to their mixing strategy
  - ConfBridge: marked users, admins, etc.
  - Parking: which parking lot am I in
  - Stasis: ARI specific logic to know who controls a channel
  - Agent waiting pools: how long has a Queue agent been waiting for a call
- Solution: give a bridge a 'class' with callbacks to update their application state
- ConfBridge: still prohibited


bridge_agent_hold
ast_bridge_basic
bridge_stasis
ast_bridge_parking
bridge_base
ast_bridge_features
features
ast_bridge_channel
ast_channel *chan
1
channels
0..*
ast_bridge
LIST(ast_frame) action_queue
int allowed_capabilities
v_table
ast_bridge_methods
void (*destroy)(ast_bridge *self)
void (*dissolving)(ast_bridge *self)
int (*push)(ast_bridge *self, ast_bridge_channel *bc, ast_bridge_channel *swap)
int (*pull)(ast_bridge *self, ast_bridge_channel *bc)
void (*notify_masquerade)(ast_bridge *self, ast_bridge_channel *bc)
int (*get_merge_priority)(ast_bridge *self)
int (*push_peek)(ast_bridge *self, ast_bridge_channel *bc, at_bridge_channel *swap)
technology
ast_bridge_technology

The Big Picture

The Current* Bridging Framework



Questions?