P802.1CQ/D0.6 Preview v04

Roger Marks
(EthAirNet Associates)
P802.1CQ Editor
2021-06-11

thanks to Antonio de la Oliva and Lily Lv
for review and constructive comments

Summary

- P802.1CQ/D0.5 was reviewed in TG Ballot.
- Comment resolution was completed in November.
  - Address Blocks were introduced for address claiming
- In March, Editor presented “Block Address Registration and Claiming (BARC)”
  - cq-marks-BARC-0321-v00.pdf
  - address blocks used for registrar-managed addresses as well
  - Address Registration and Claiming (ARC)
    - address blocks, and also claiming address ranges using MAAP
      - presented also to IEEE 1722 Working Group
- Main issues raised in March concerned VLAN operation
- This contribution previews P802.1CQ/D0.6
  - refinements and details since March presentation
  - discussion on improved VLAN support
  - v01 presented to TSN at May 802.1 Interim; this version (v03) adds detail

BARC assigns MAC Addresses in Address Blocks

1) Address Blocks (ABs) are sets of local addresses.
2) An AB includes equal-sized unicast and multicast address subblocks.
3) No BARC address falls within more than one AB.
4) An Address Block Designation (ABD) is a CABA or a RABI.
5) Claimable AB Address (CABA) is claimable by a Claimant without using a Registrar.
    - identifies Claimable Address Blocks (CABs) holding Claimable Addresses (CAs)
    - CABA is a multicast MAC address, not in any AB, and used as a DA.
6) RABI
    - identifies a Registrable Address Block (RAB) holding Registrable Addresses (RAs)
    - Registrable Address Block Indicators (RABIs): held in inventory of a Registrar
        - may be assigned to Claimants
        - may be claimed by Registrants
7) A large set of Temporary Unicast Addresses (TUAs) is specified
    - useful for initial discovery by Claimant lacking a unicast address

MAC Address Categorization

determinable via inspection:	**Expanded name**	**I/G**	**indicates, by inspection**
CA [ICA=unicast GCA=multicast]	claimable address, in claimable address block (CAB)	U,M	CABA, CAB Size, CAB (including all other CAs in CAB)
CABA	CAB Address	M	CAB Size, CAB
RA [IRA=unicast GRA=multicast]	registrable address, in registrable address block (RAB)	U,M	BABI Size [Basic ABI Size]
TUA	temporary unicast address	U	note: ~6.9E10 to choose among

Address Block Designation (ABD) Categorization

ABD	Address Block Designation (CABA or RABI)	not an address	AB (including all RAs in AB)
RABI	RAB Identifier	not an address	RAB Size, BABI Size, RAB, MABI Size [Multiple ABI Size]
CABA	CAB address	M	CAB Size, CAB

BARC MAC Address Structure

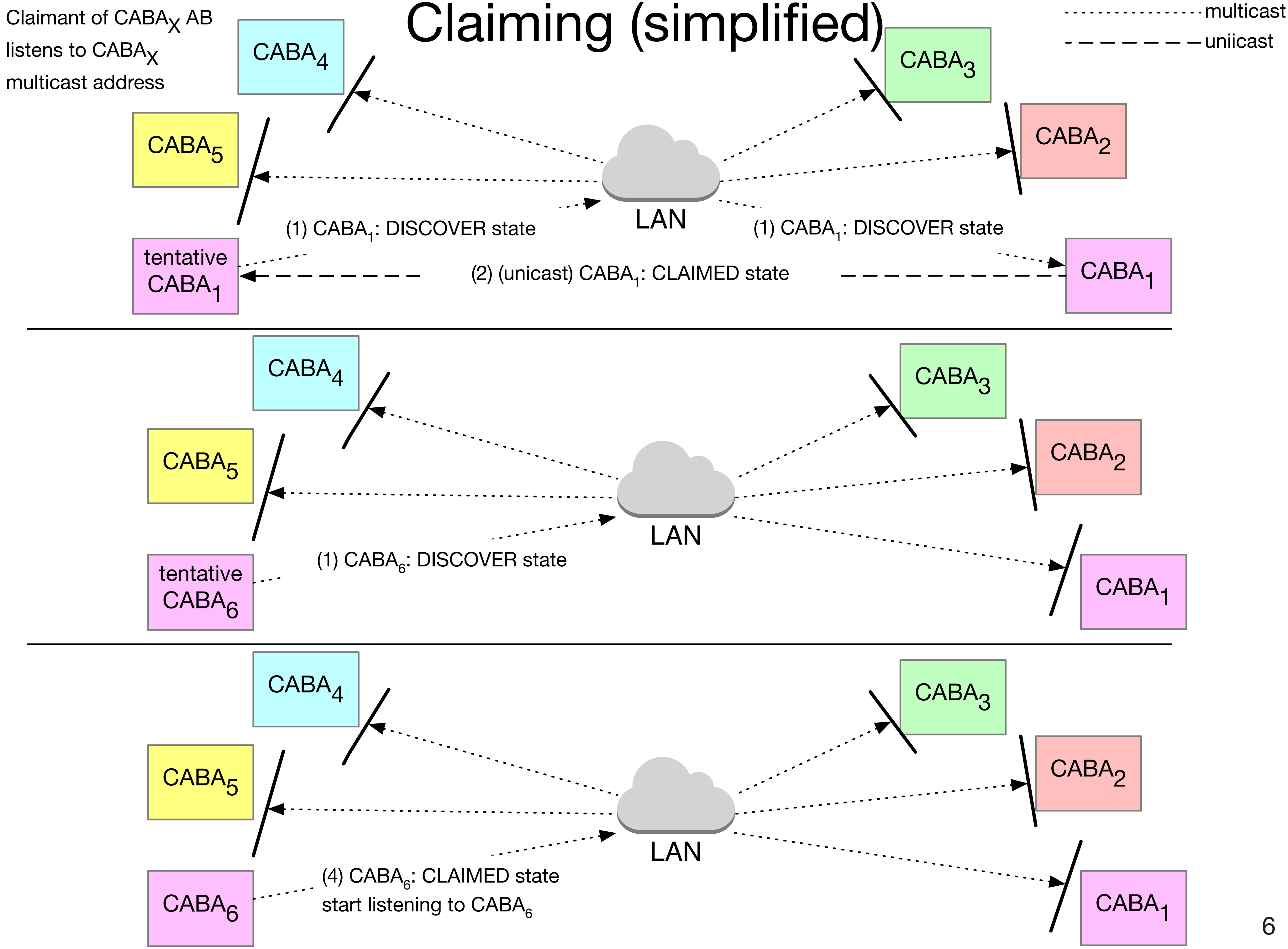
N11	r	i	j	k	for registrable addresses, r=1; for claimable addresses, r=0
N10	1	1	1	m	m is the usual multicast (I/G) bit; 111 is local "SAI" range per IEEE Std 802c
N9					0000 for claimable addresses
N8					
N7					
N6					
N5					
N6					
N5					
N2					
N1					
N0					

12 nibbles per 48-bit address

- address block includes subblocks of
  - 16^{jk} claimable addresses, or
  - 16^{jk} registrable addresses (or aggregated into larger blocks)
- for claimable addresses, *i* distinguishes
  - Claimable Addresses (CAs) from
  - CABAs
    - identifiers that are also used as addresses
- see Appendix for details

	r	*i*	*jk*	*m*
CA	0	1	CAB Size	I/G
CABA	0	0		1
TUA	0	0	0	0
RA	1	RABI Option	BABI Size	I/G

Claiming (simplified)



Registrar

- Claimant need not be aware of Registrar when initiating a claim.

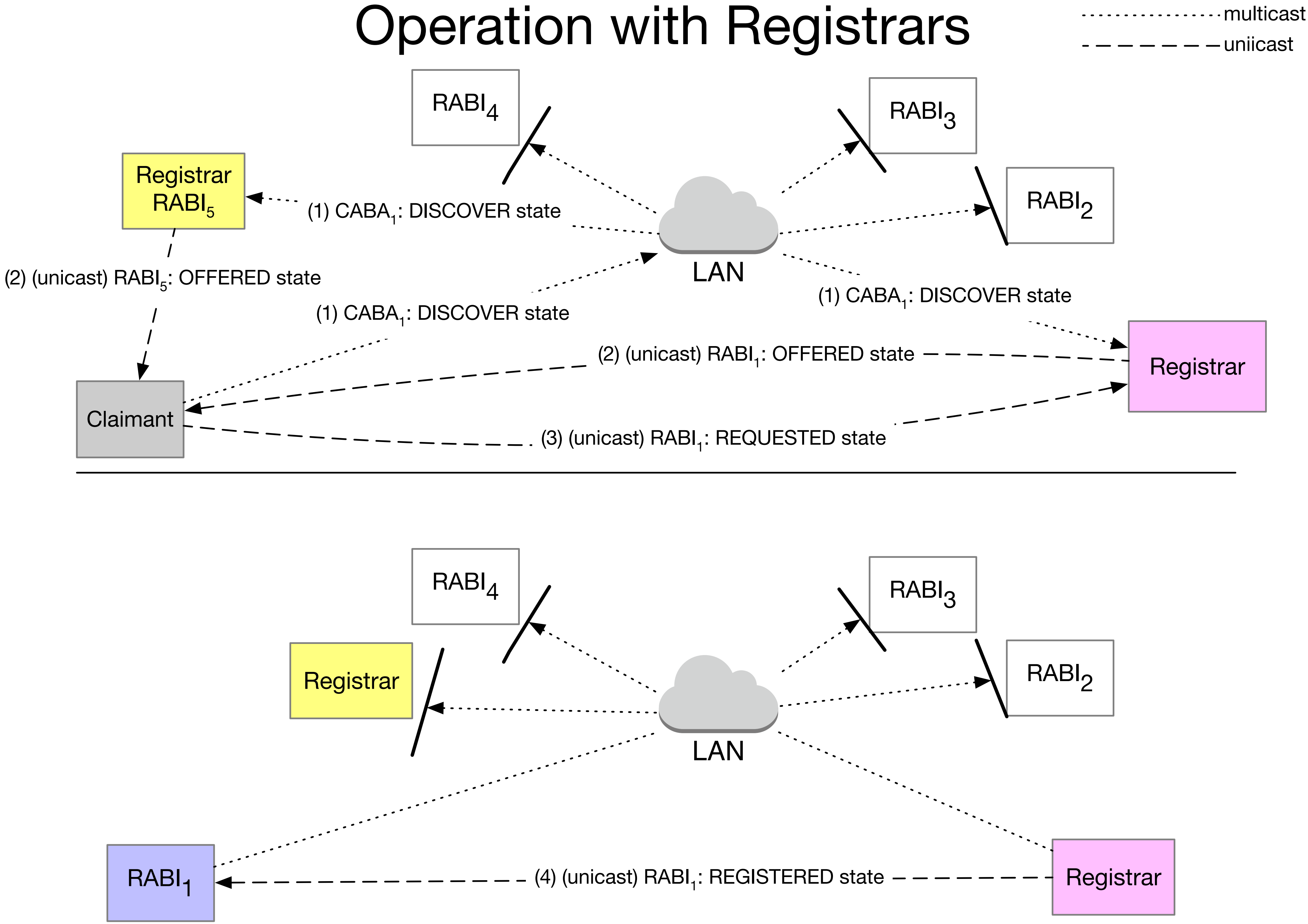
- Registrar maintains an inventory of RABIs.
  - a protocol specifies how Registrars acquire RABIs.
  - set of RABs is disjoint from the set of CABs
    - AB is either claimable (CAB) or registrable (RAB); not both

- Registrar listens for all messages to a CABA.
  - r=0, i=0, m=1, i.e. DA begins 00**-1111
    - [MMRP NumberOfValues field is 13 bits]

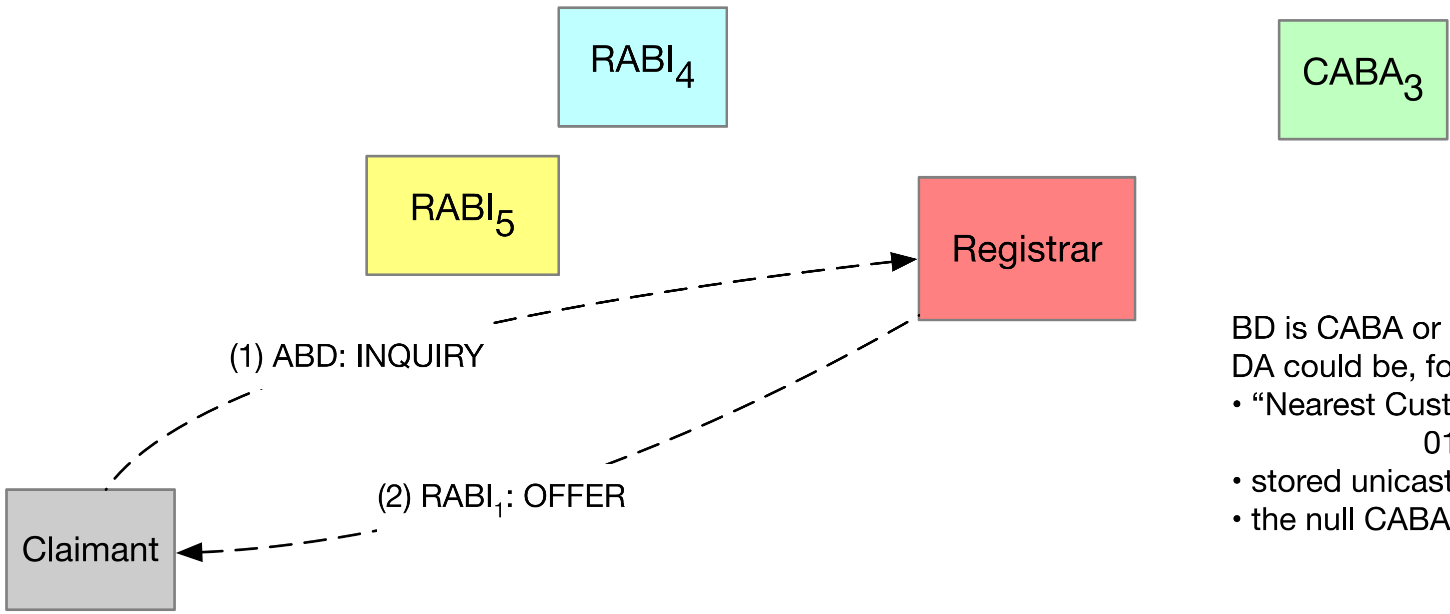
- Registrar can respond to a DISCOVER with an offer of a RABI in its inventory.
  - The offer can also defend the DISCOVER's CABA.
  - Registrar confirms registration of request for offered RABI.

- Pre-claim Inquiry lets Claimant reach Registrar or Advisor.
  - Client can learn of Registrars and received Claim proposals.

Operation with Registrars

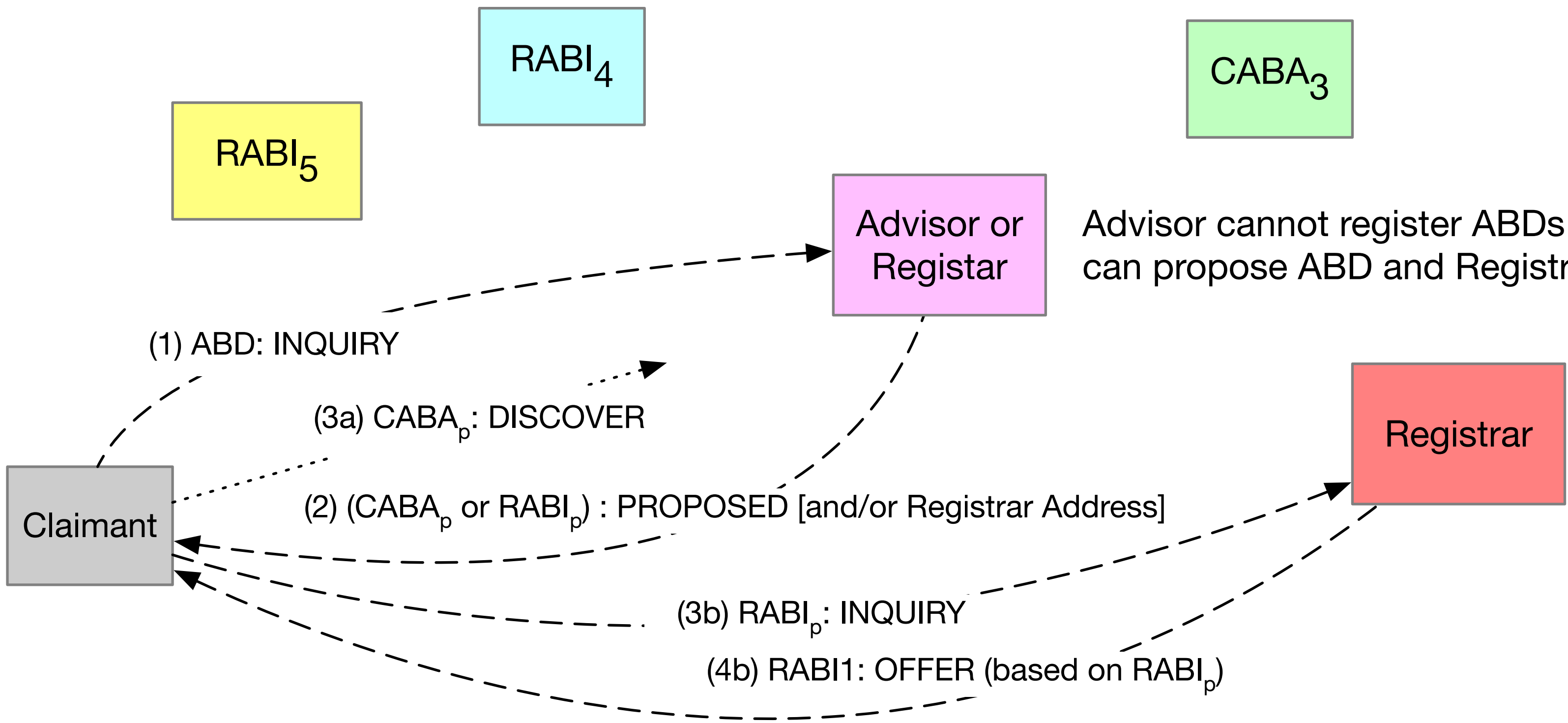


Inquiry to (anticipated) Registrar or Advisor



multicast
uniicast

BD is CABA or RABI
DA could be, for example:
• “Nearest Customer Bridge” (NCB) address
01-80-C2-00-00-00 (non-forwarding)
• stored unicast Registrar Address
• the null CABA (CABA$_0$)



Advisor cannot register ABDs;
can propose ABD and Registrar address

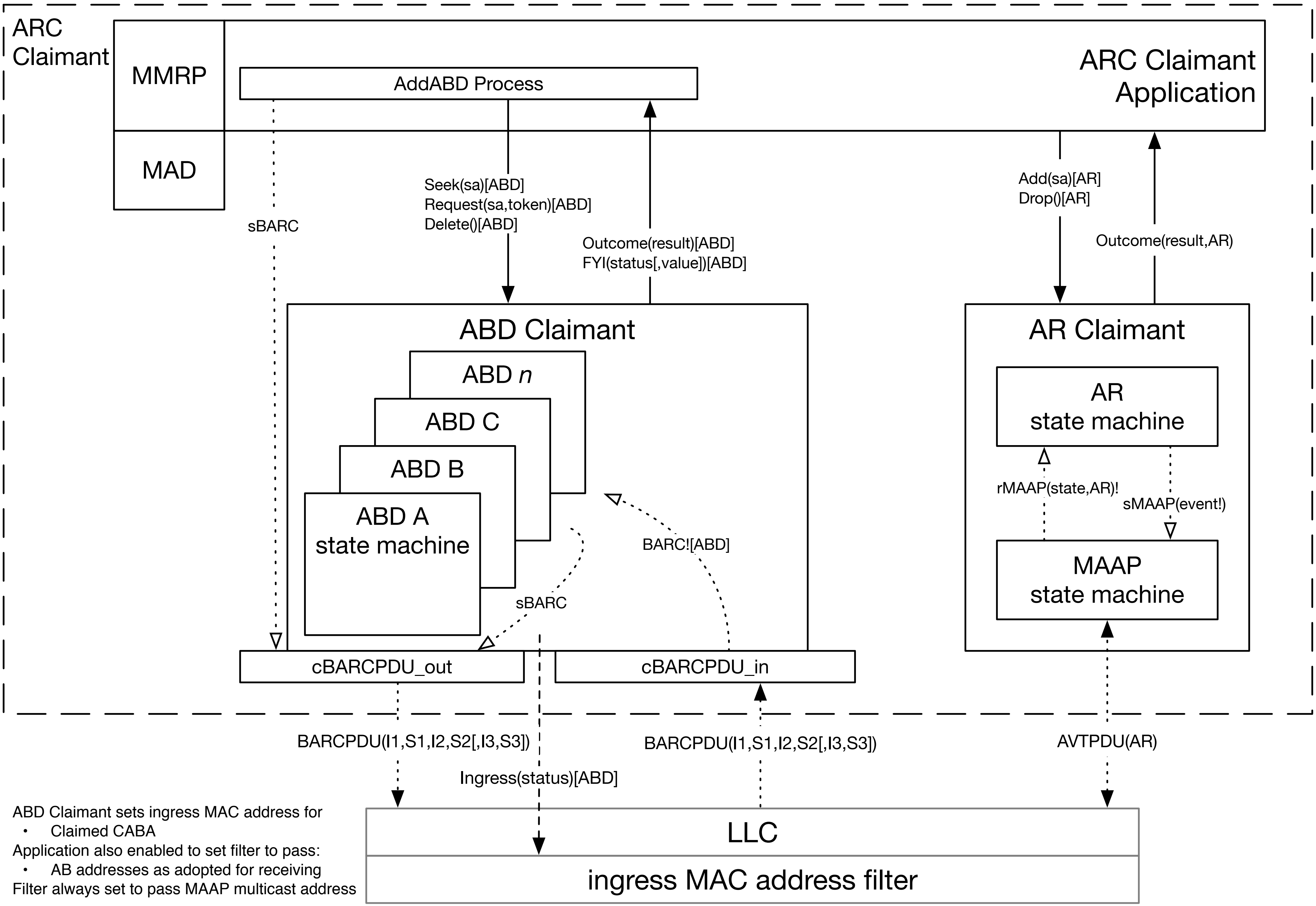
BARC Design

- A BARC architecture follows, with details including state machines.
  - additional details in Appendix

- BARC (Block Address Registration and Claiming) is put into the broader context of Address Registration and Claiming (ARC), which supports both:
  - address blocks (ABs), identified by Address Block Identifiers (ABIs)
  - address ranges (ARs), excluding addresses specified by BARC

- ARC is the general protocol
  - BARC handles ABI Registration and CABA Claiming
  - existing MAAP handles AR Claiming

ARC Architecture – ARC Claimant



ABD Claimant sets ingress MAC address for

- Claimed CABA

Application also enabled to set filter to pass:

- AB addresses as adopted for receiving

Filter always set to pass MAAP multicast address

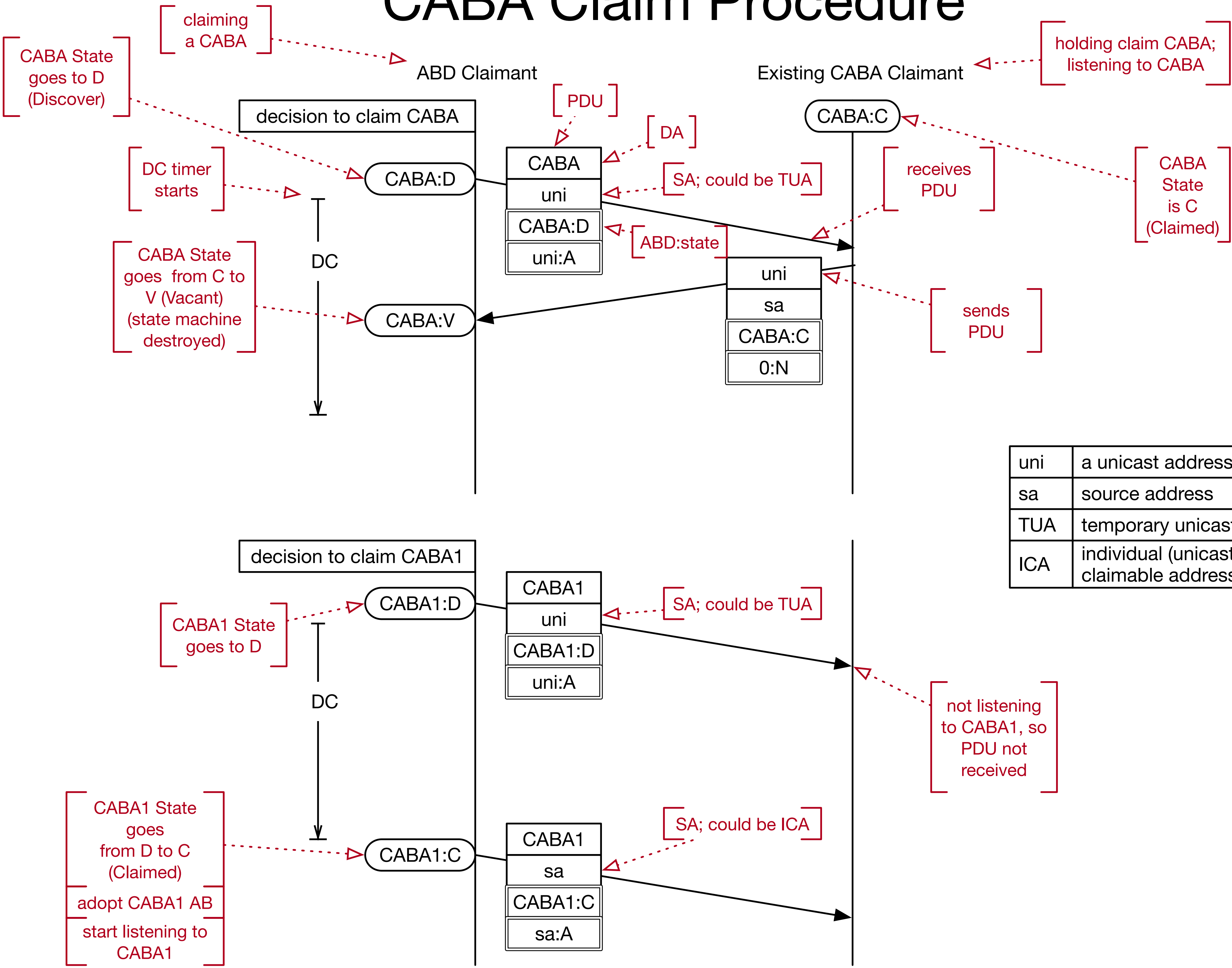
BARCPDU Summary

field name	purpose	content
DA	dest addresss	DA
SA	source address	
E	Ethertype	[tbd; could be 22F0 (MAAP Ethertype)]
t	subtype	[tbd, per 1722 WG; see IEEE 1722 Table 6]
S1	State 1	D (Discover), C (Claimed), V (Vacant), R (Registered), I (Inquiry), P (Proposal), A (address), RD, RC, RV, RX, N(null)
I1	Identifier 2	48-bit address or ABI
S2	State 2	O (Offered), A (address), N (null)
I2	Identifier 2	48-bit address or ABI
S3	State 3	A (address), T (token)
I3	Identifier 3	48-bit address or token

AVTPDU Summary

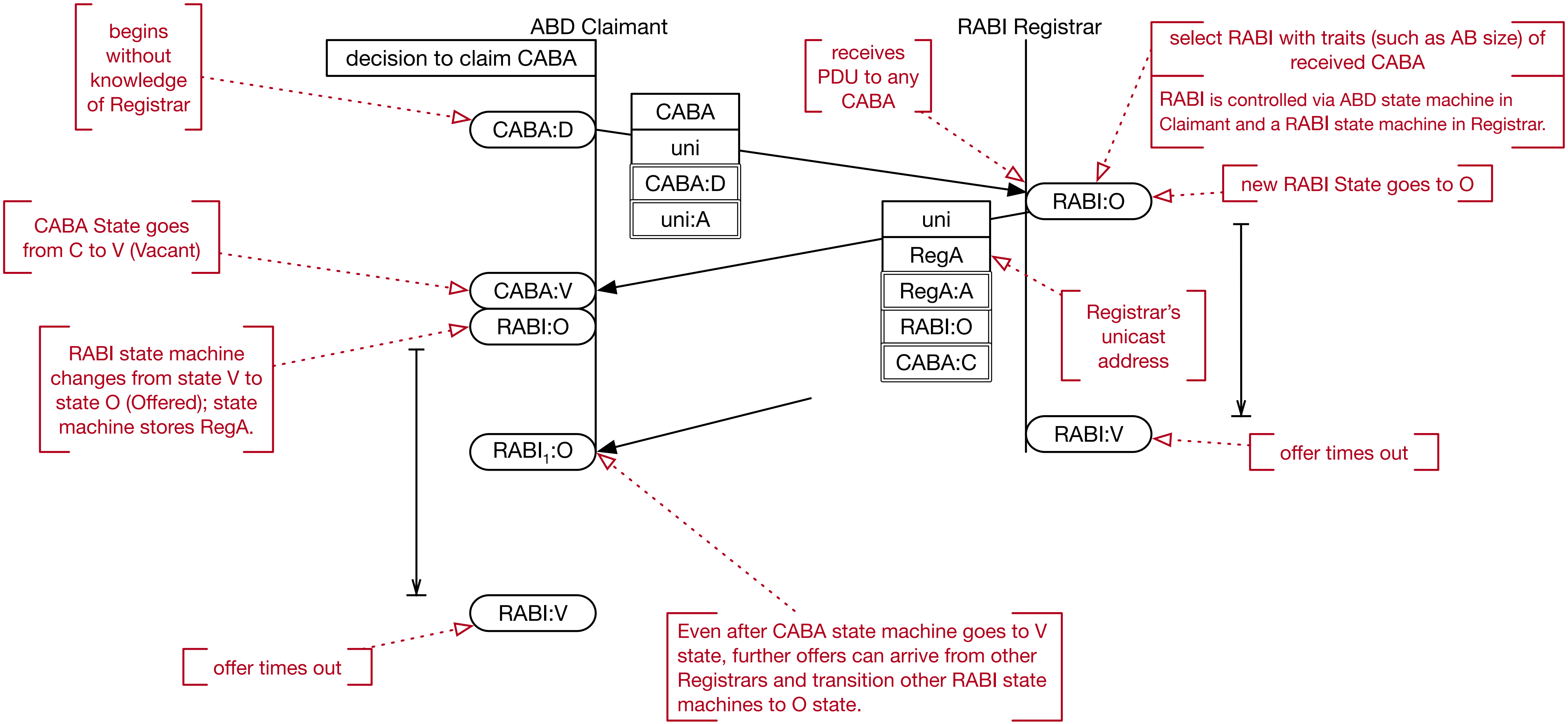
field name	purpose	content
DA	dest addresss	91:E0:F0:00:FF:00 for MAAP multicast
E	Ethertype	22F0 (MAAP Ethertype)
t	subtype	FE per IEEE 1722 Table 6

CABA Claim Procedure

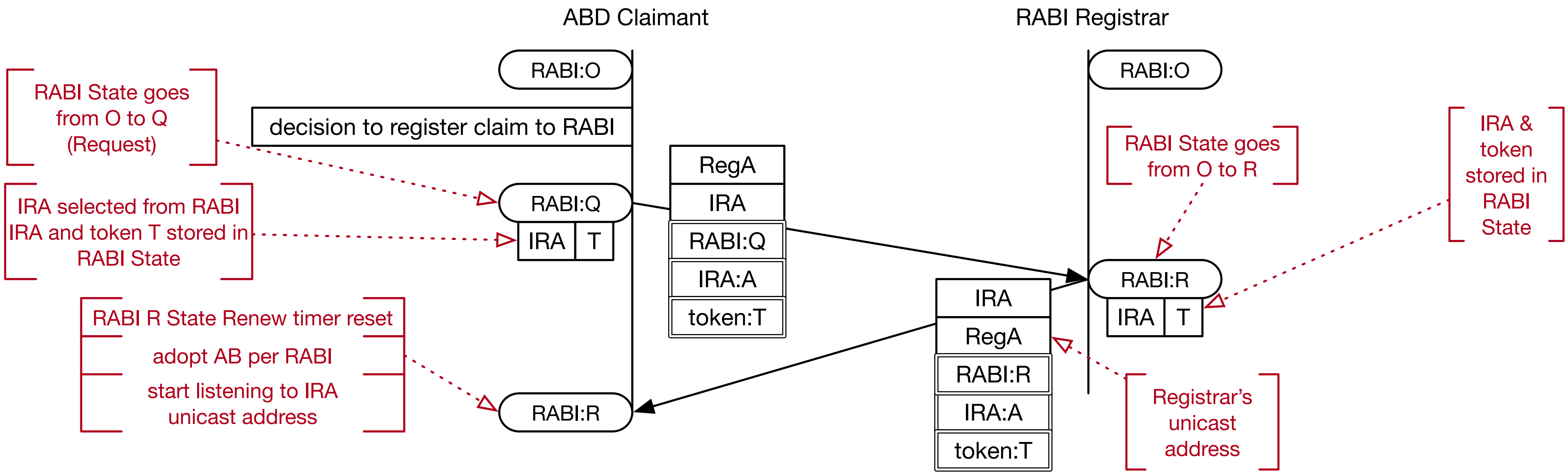


uni	a unicast address
sa	source address
TUA	temporary unicast address
ICA	individual (unicast) claimable address

ABD Claimant/Registrar Procedure

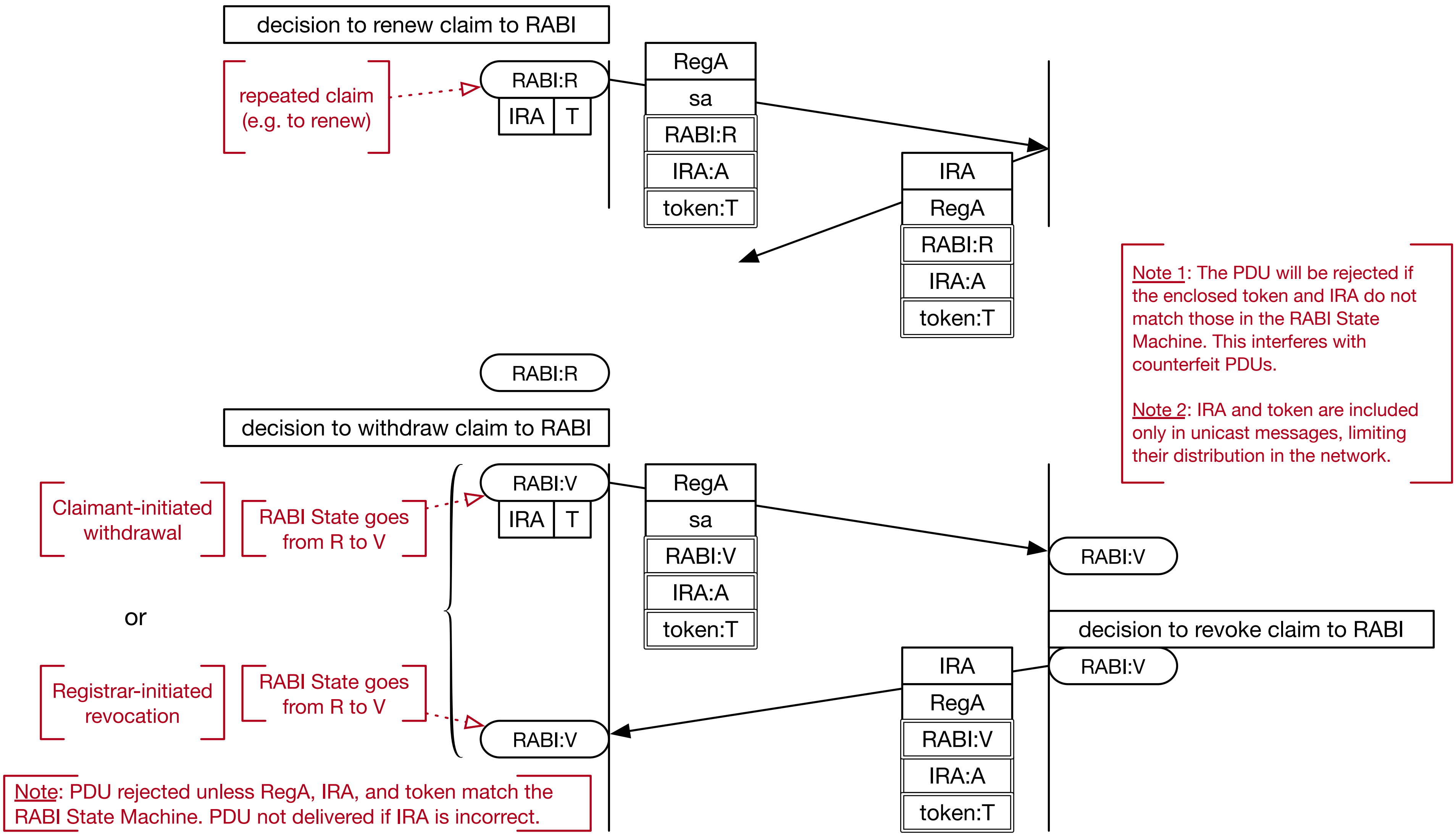


RABI Registration



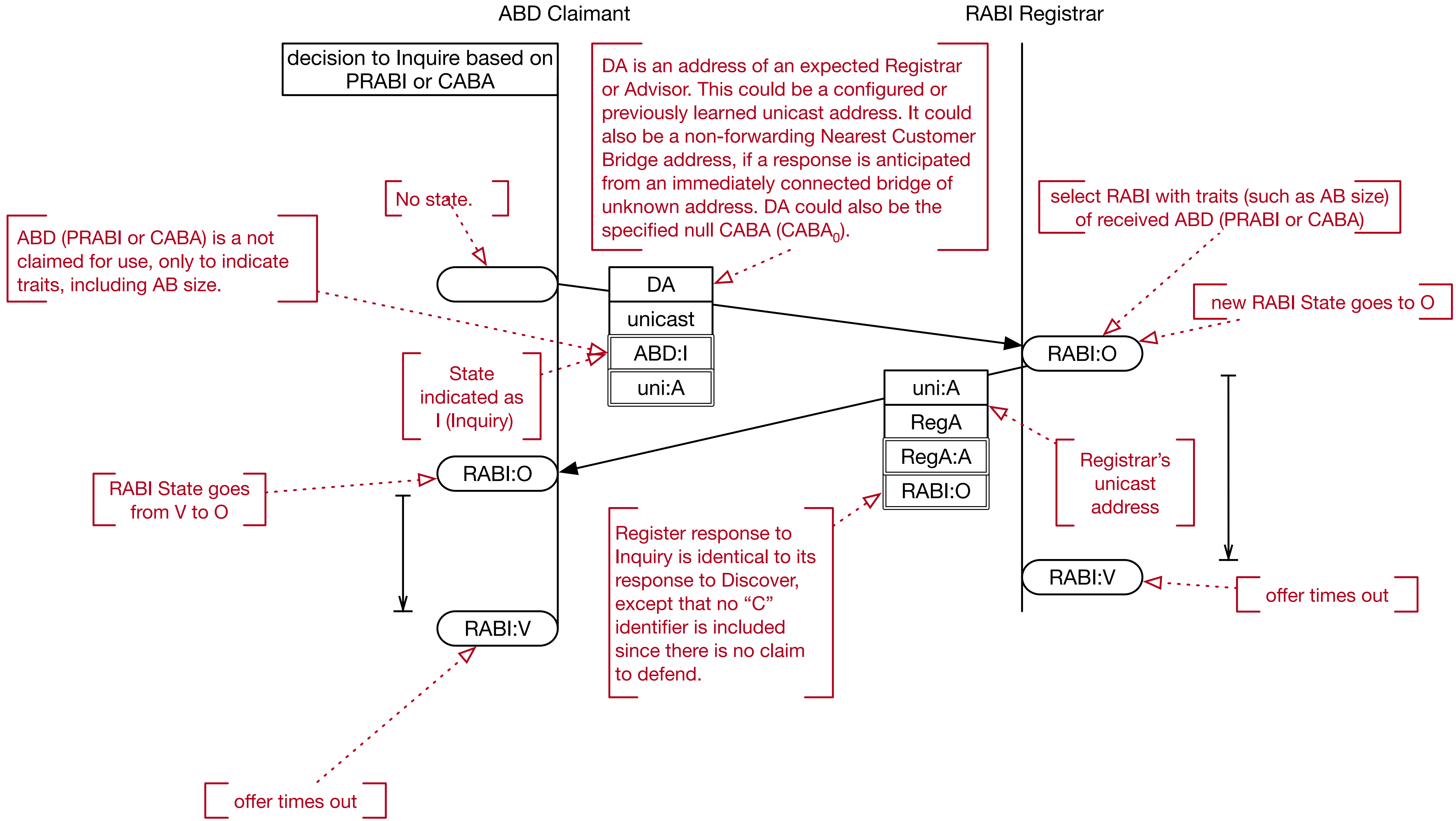
IRA	individual (unicast) registrable address

Renewal and Withdrawal of a Registration

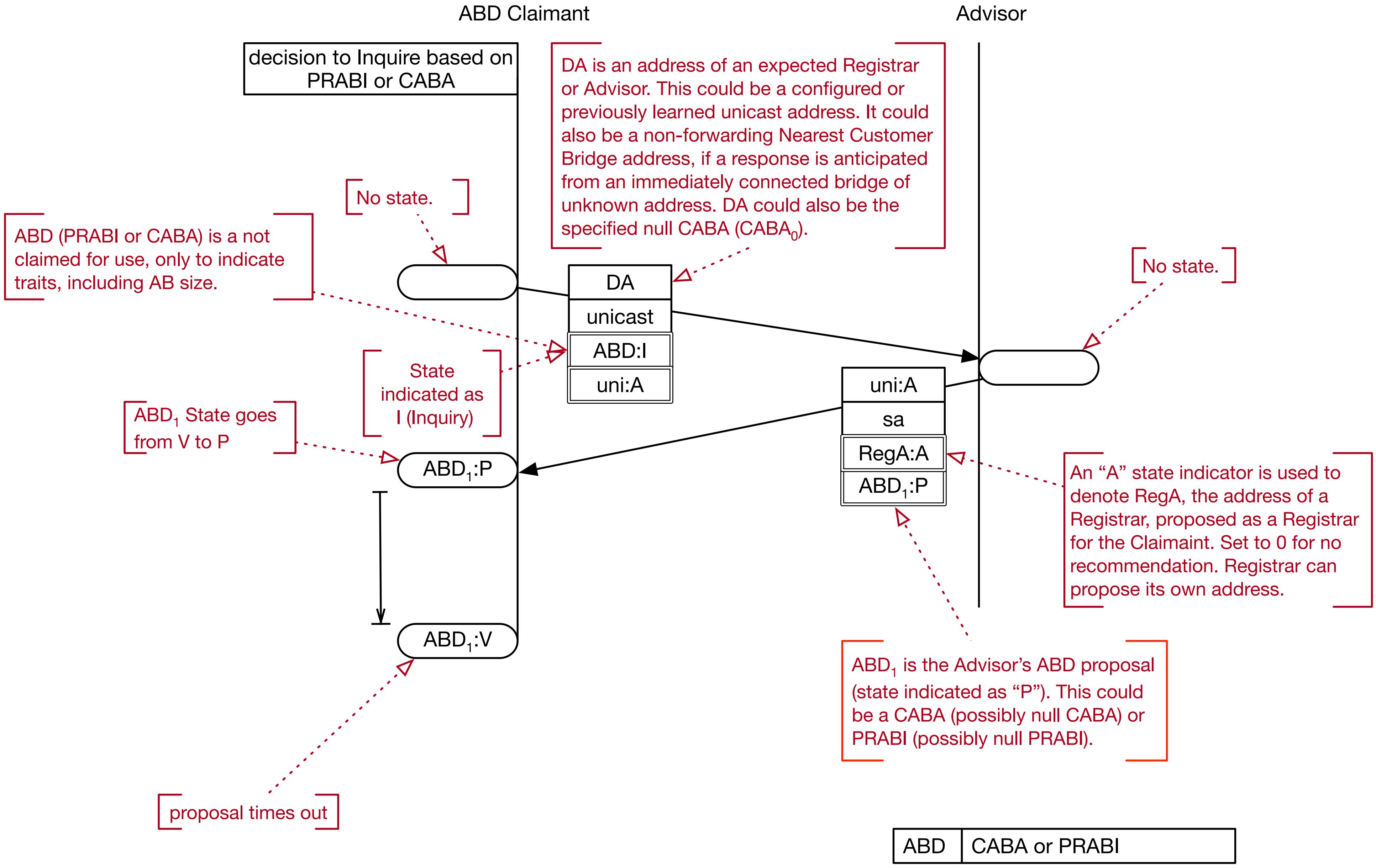


IRA	individual (unicast) registrable address

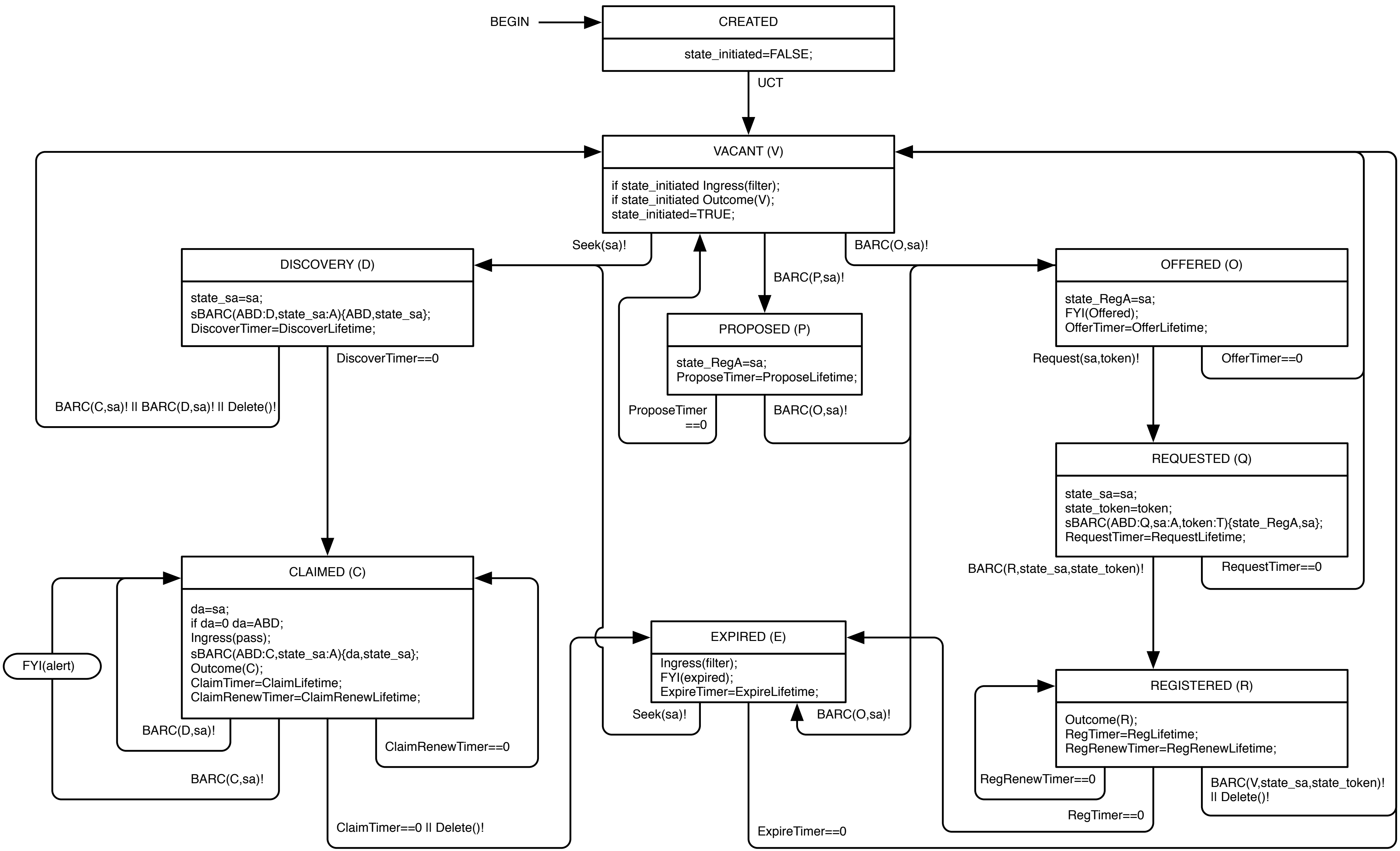
Inquiry followed by Registrar Offer



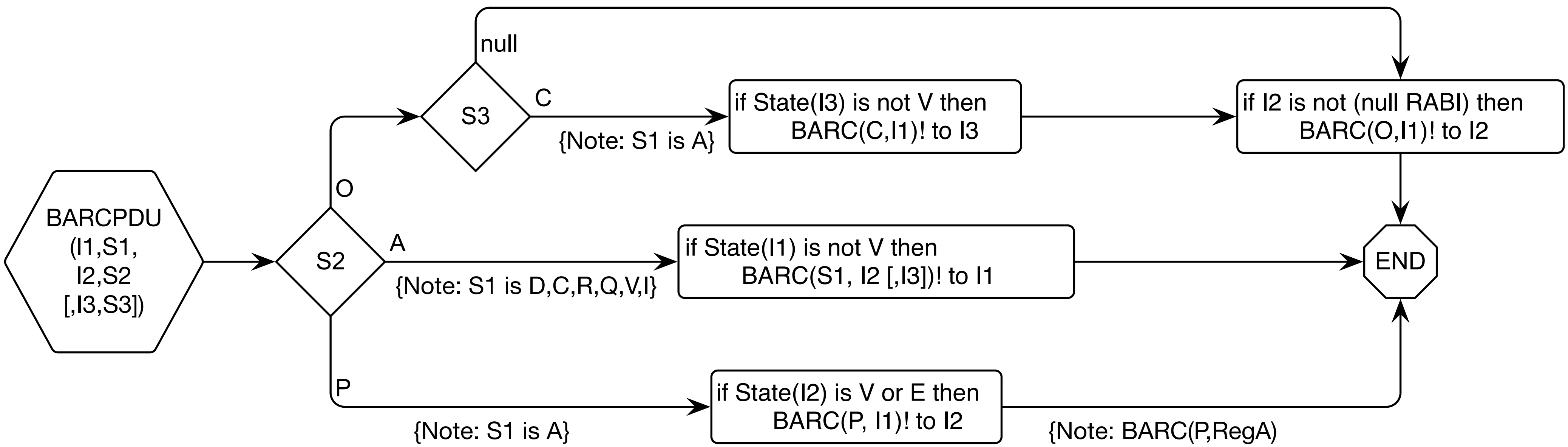
Inquiry followed by Advisor Proposal



ABD Claimant: ABD State Machine

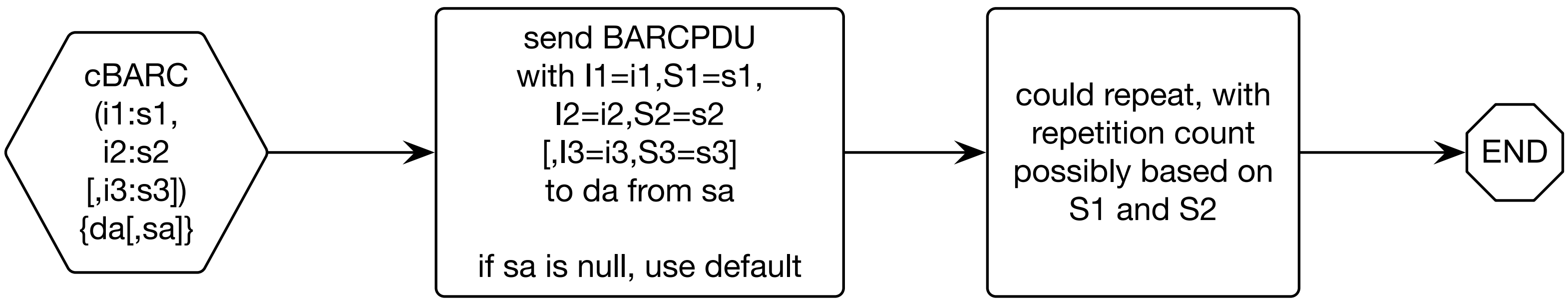


ARC Claimant: cBARCPDU_in

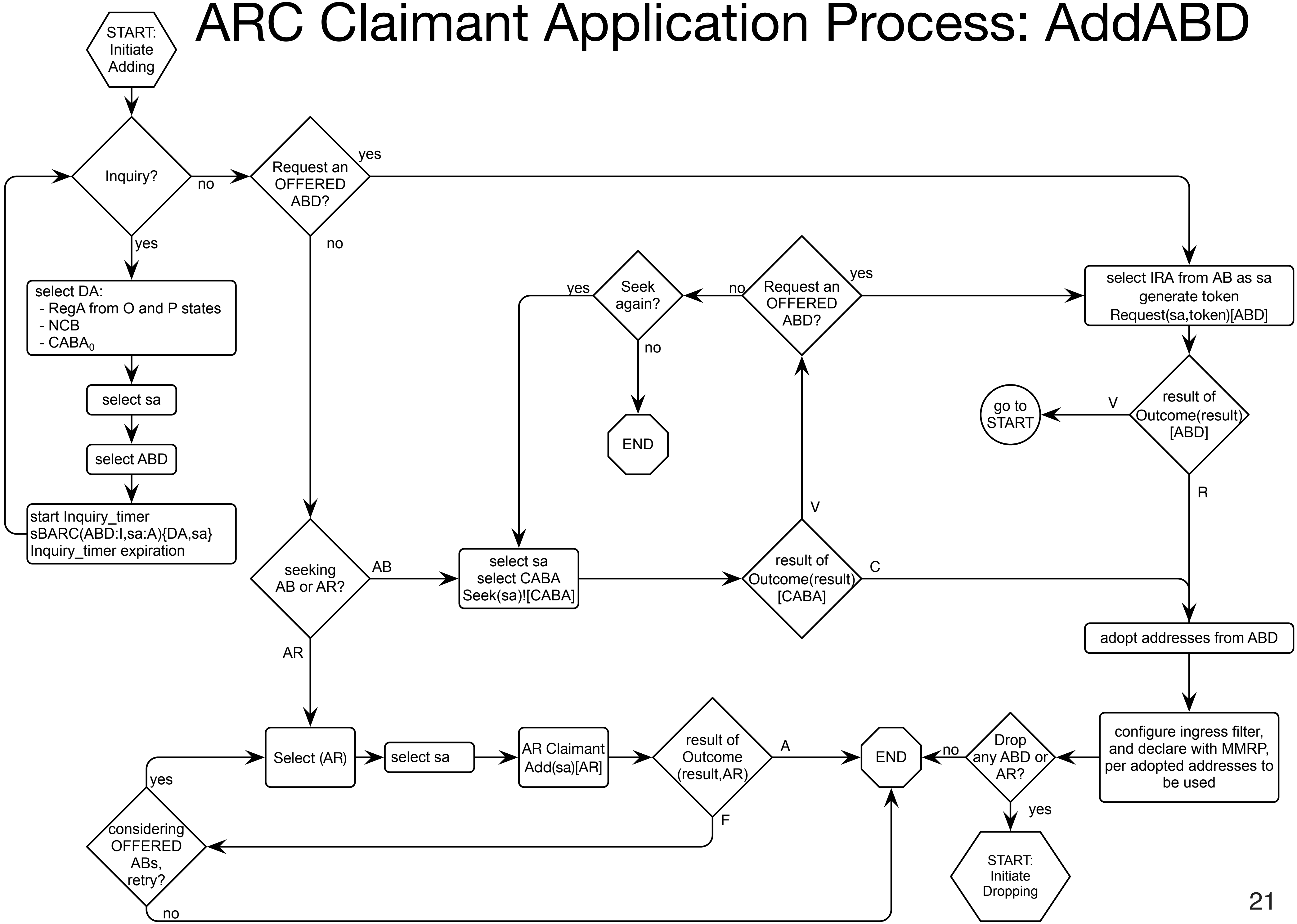


{Note: square-bracketed parameters are sometimes absent.}

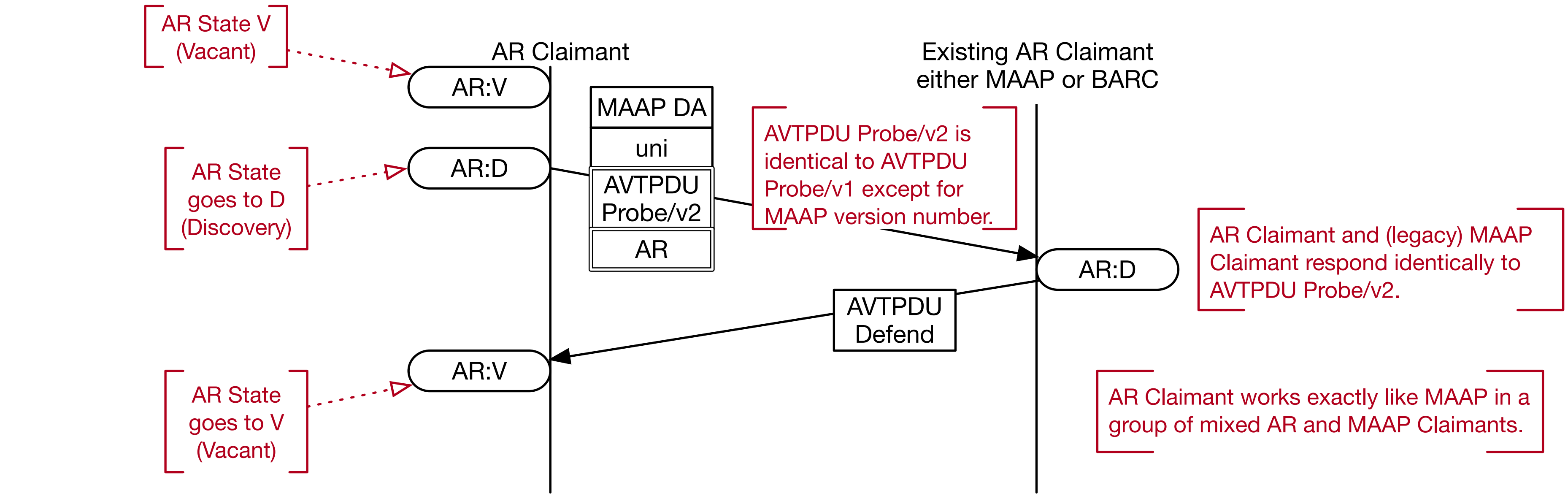
ARC Claimant: cBARCPDU_out

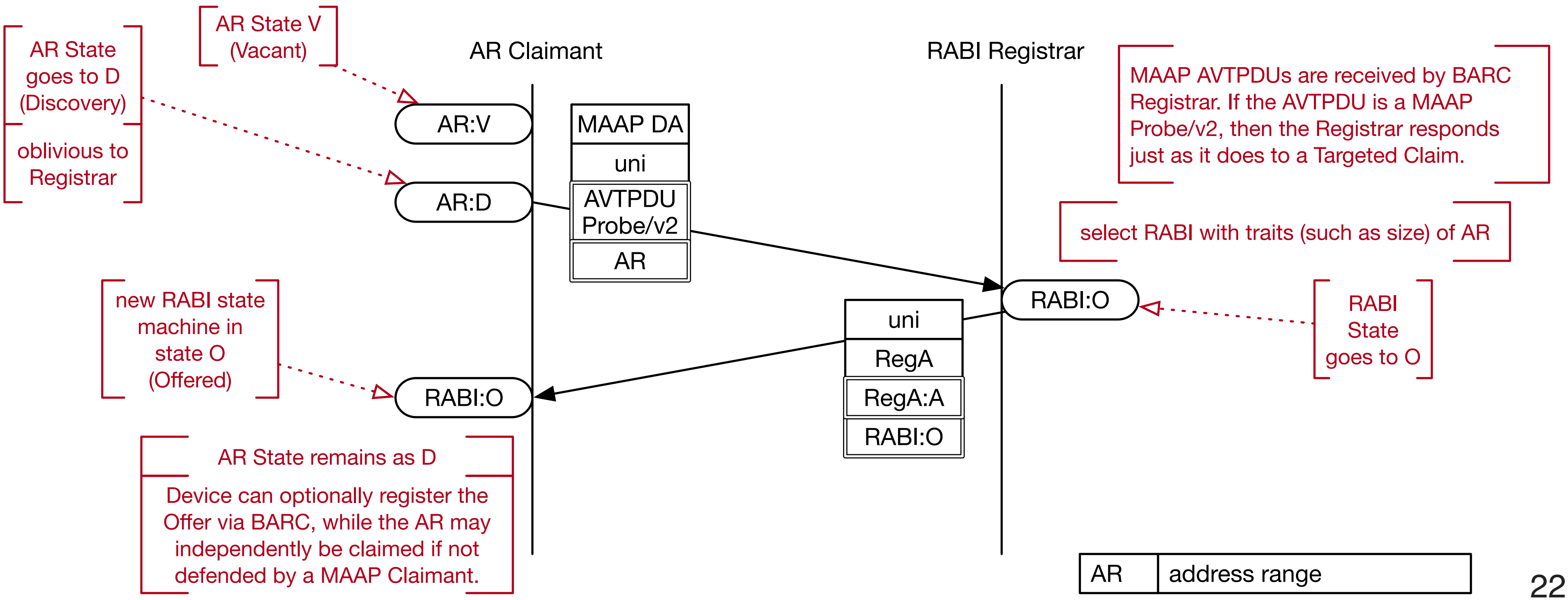


ARC Claimant Application Process: AddABD

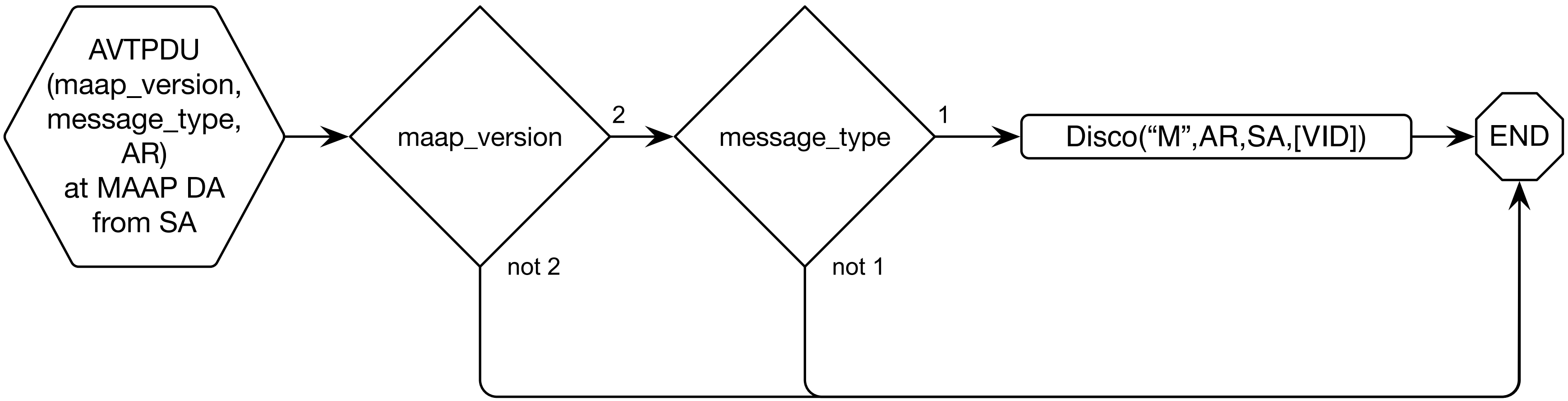


AR Claimant Procedure





BARC Registrar: AVTPDU Processor



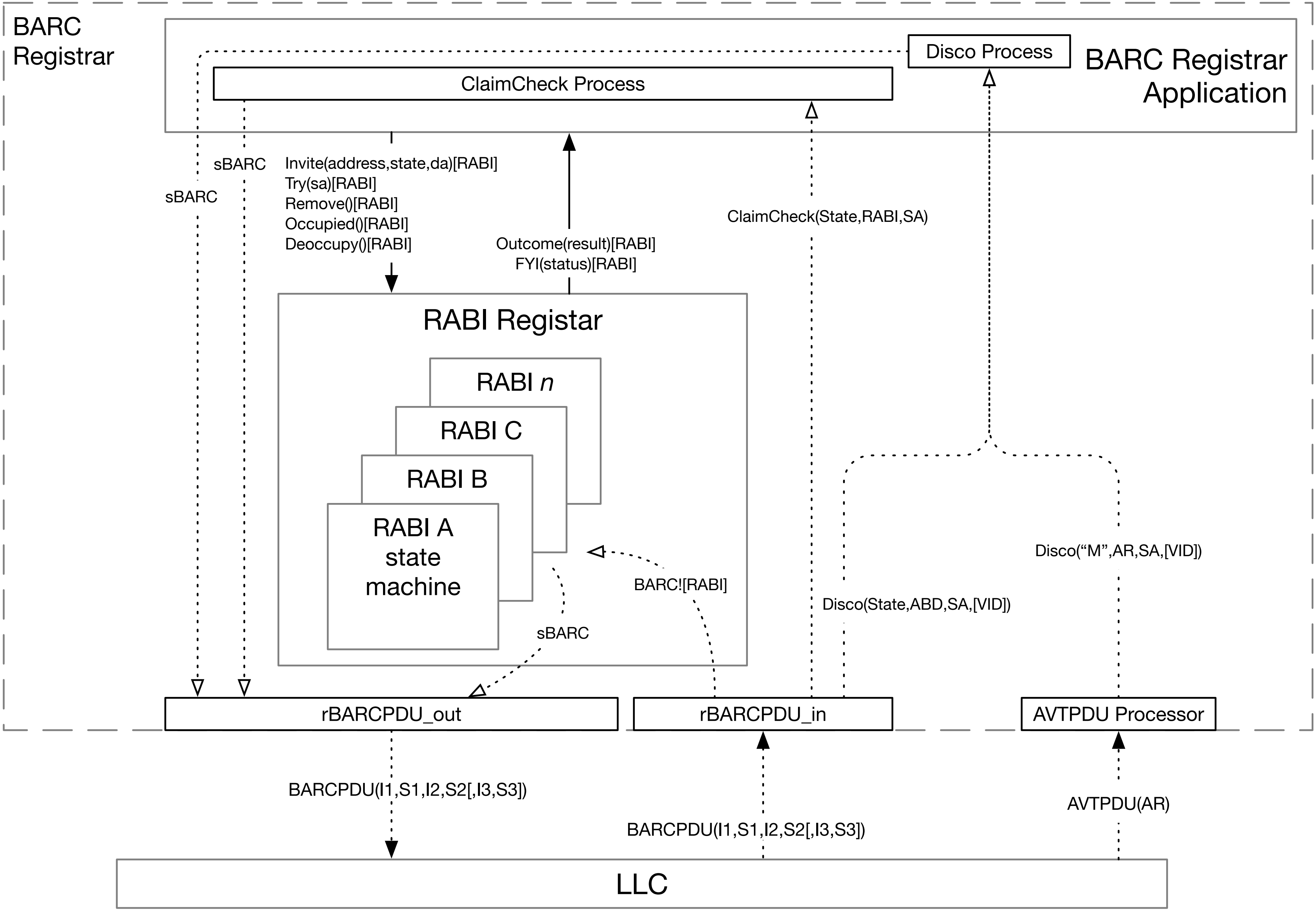
AR State Transition Table

	State		
Event	**VACANT (V)**	**DISCOVERY (D)**	**ACQUIRED (A)**
Add(sa)!	sMAAP(Begin(AR,sa)!) **DISCOVERY**		
rMAAP(AR:Defend)!		Outcome(A,AR) **ACQUIRED**	
rMAAP(AR:Initial)!		Outcome(F,AR) **VACANT**	Outcome(X)[AR] **VACANT**

rMAAP(AR:State!) invokes an event at the state machine when the MAAP state changes to State

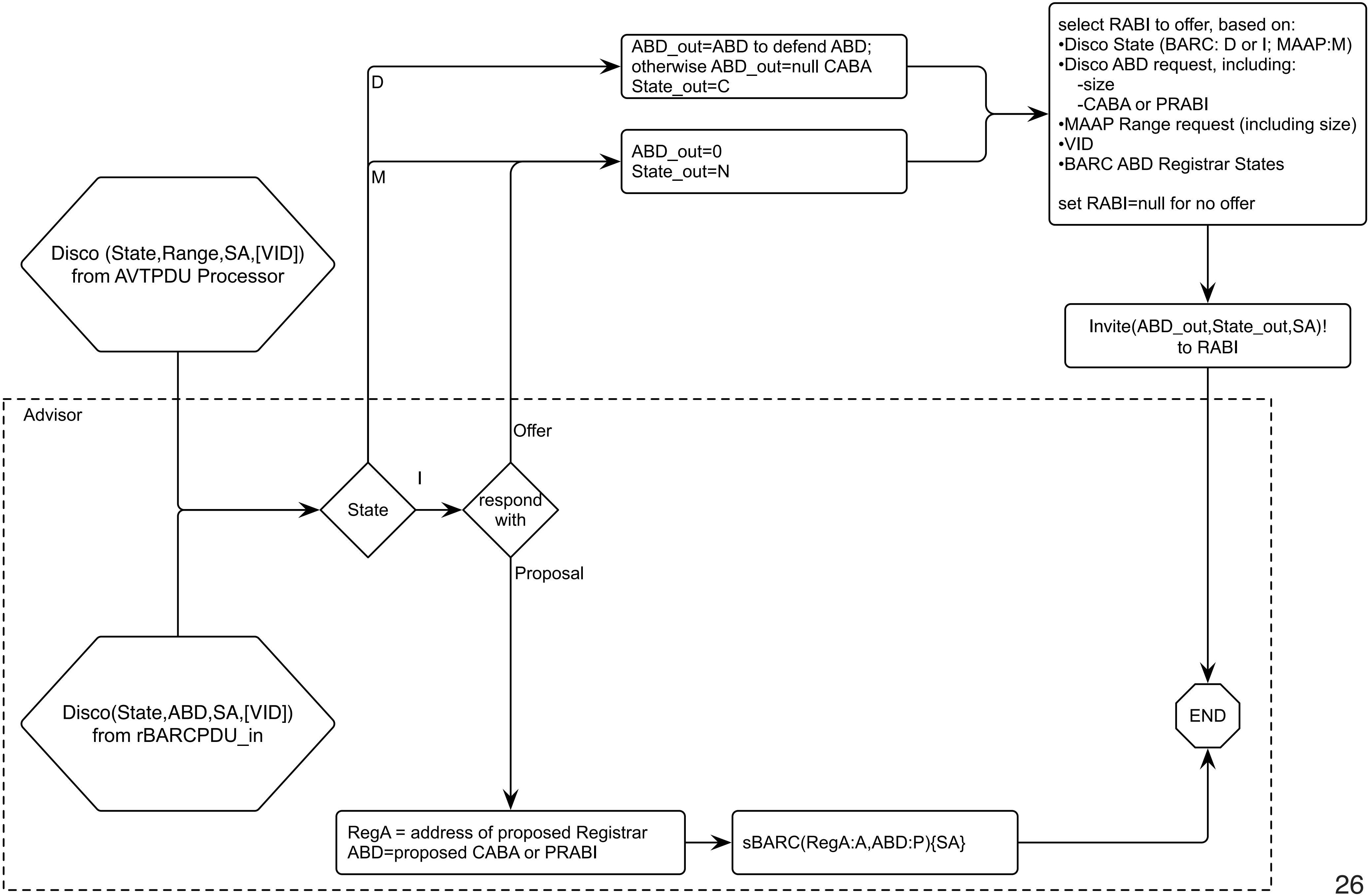
sMAAP(Action!) invokes Action! event at MAAP state machine

BARC Architecture – Registrar

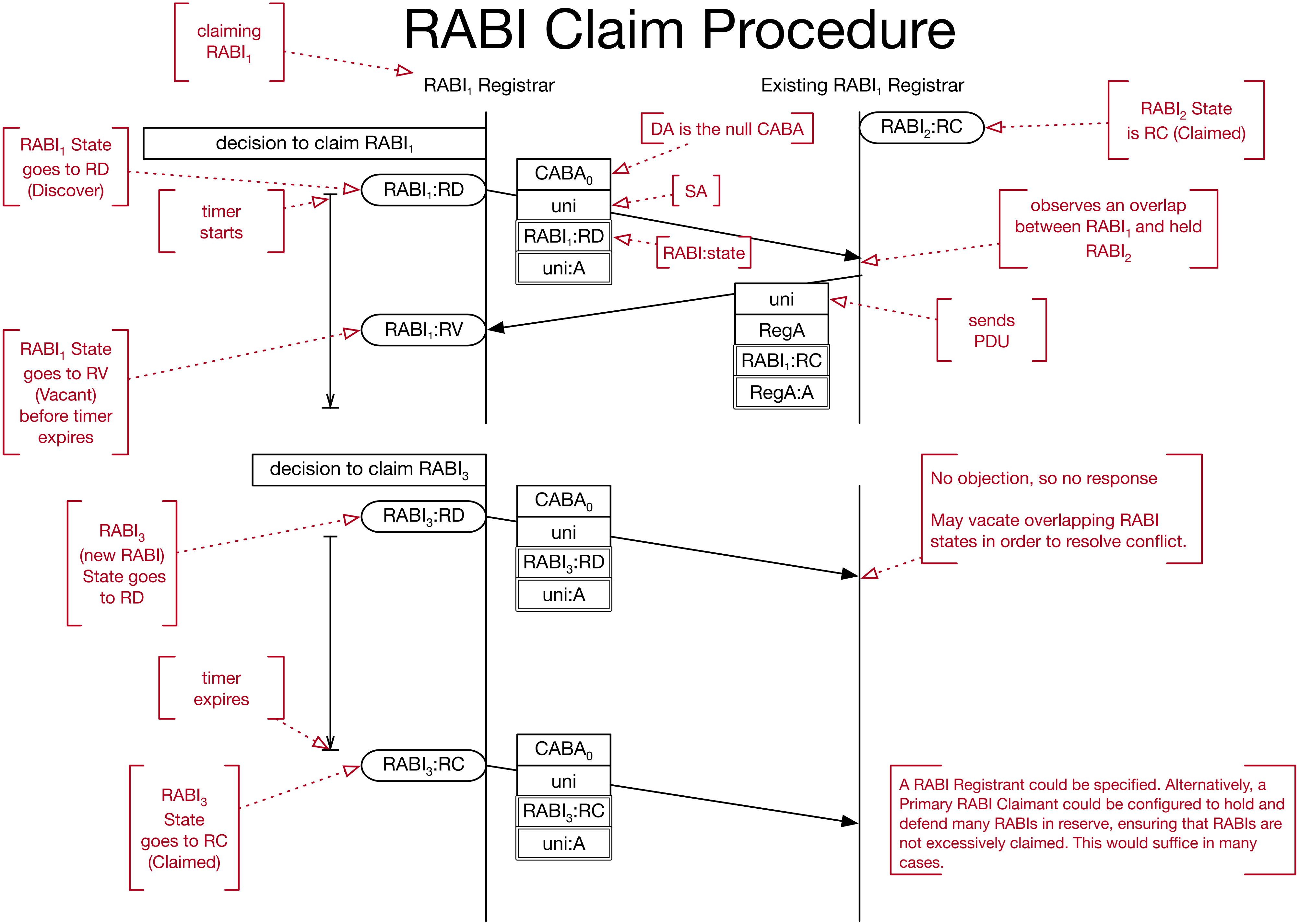


ingress MAC address filter accepts BARCPDUs addressed to any CABA
and MAAP multicast address for AVTPDUs

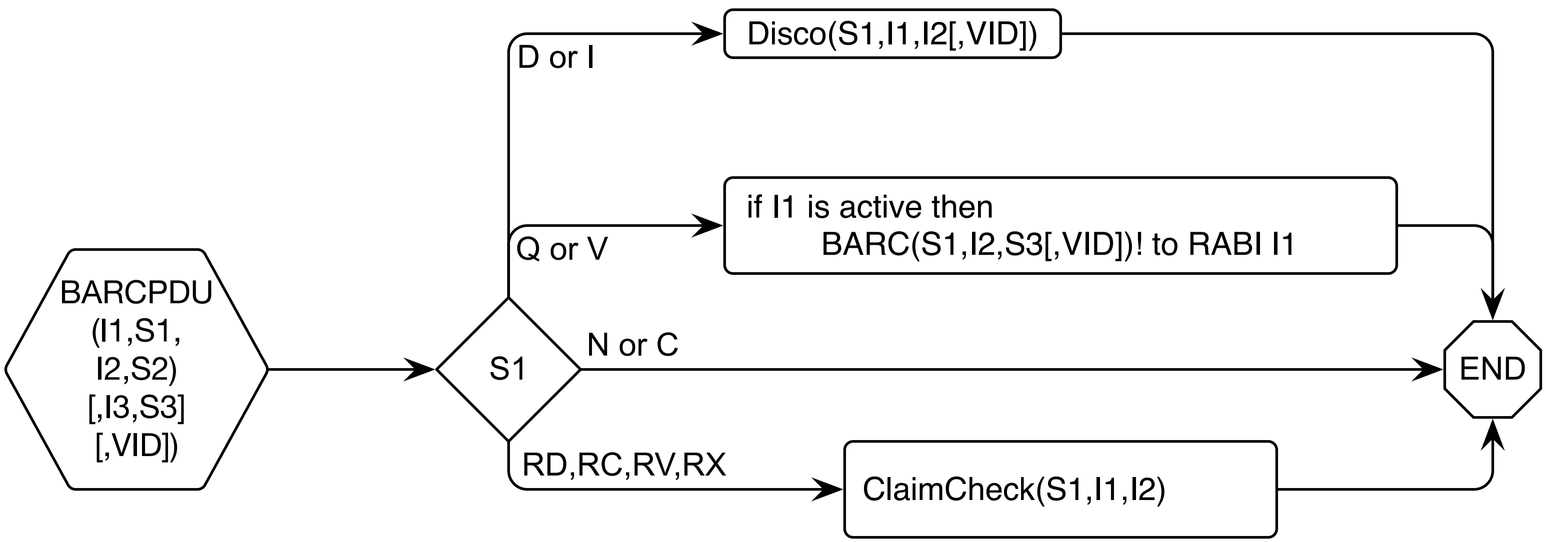
BARC Registrar Application: Disco Process



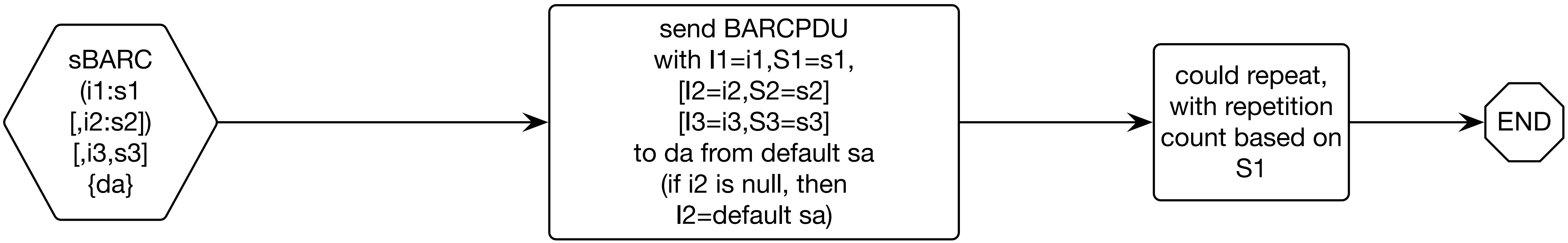
RABI Claim Procedure



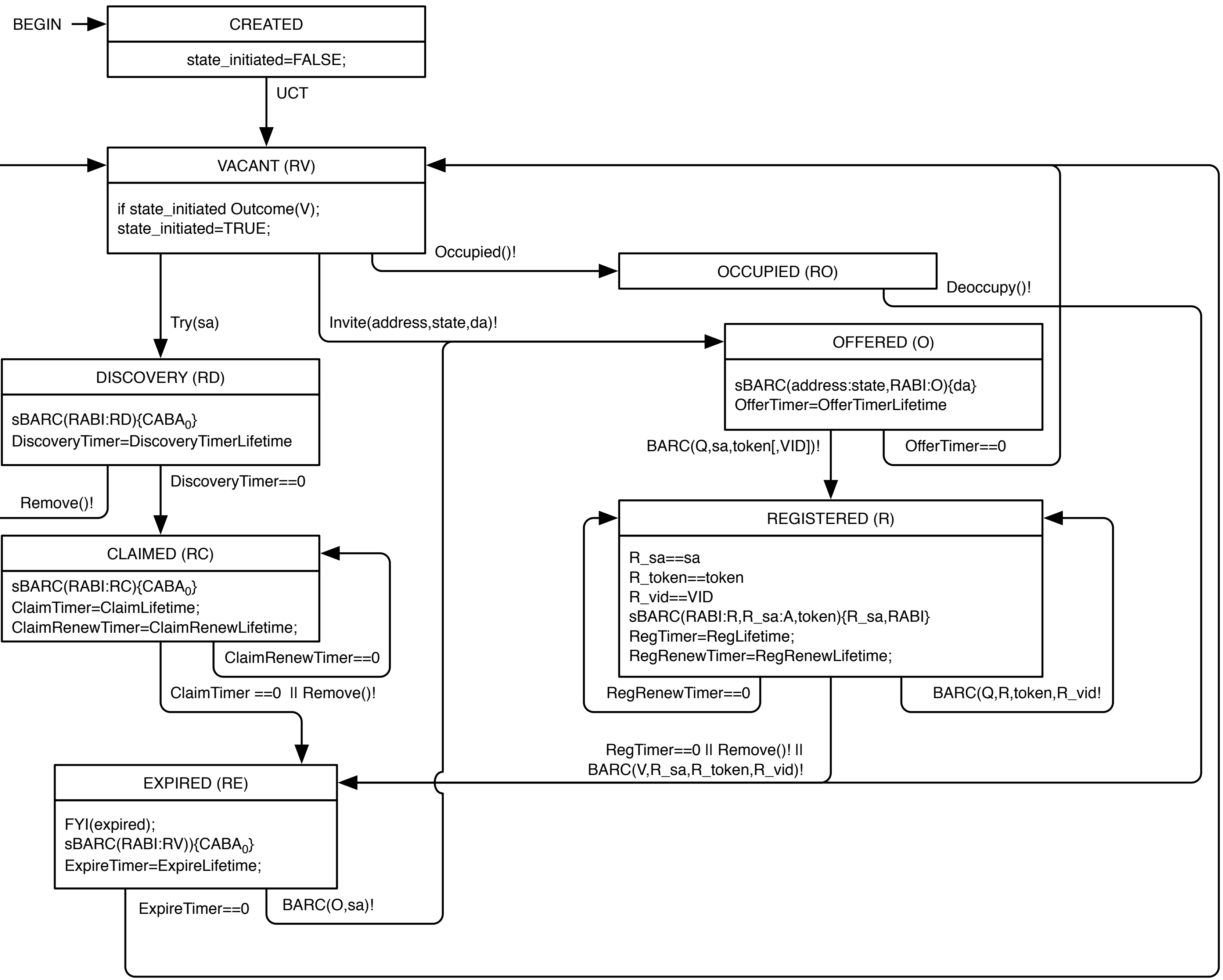
BARC Registrar: rBARCPDU_in



BARC Registrar: rBARCPDU_out



Registrar: RABI State Machine



VLANs

- All address assignments are specific to the VLAN (or VLANs) in which messaging is communicated and under which the assignment was completed.

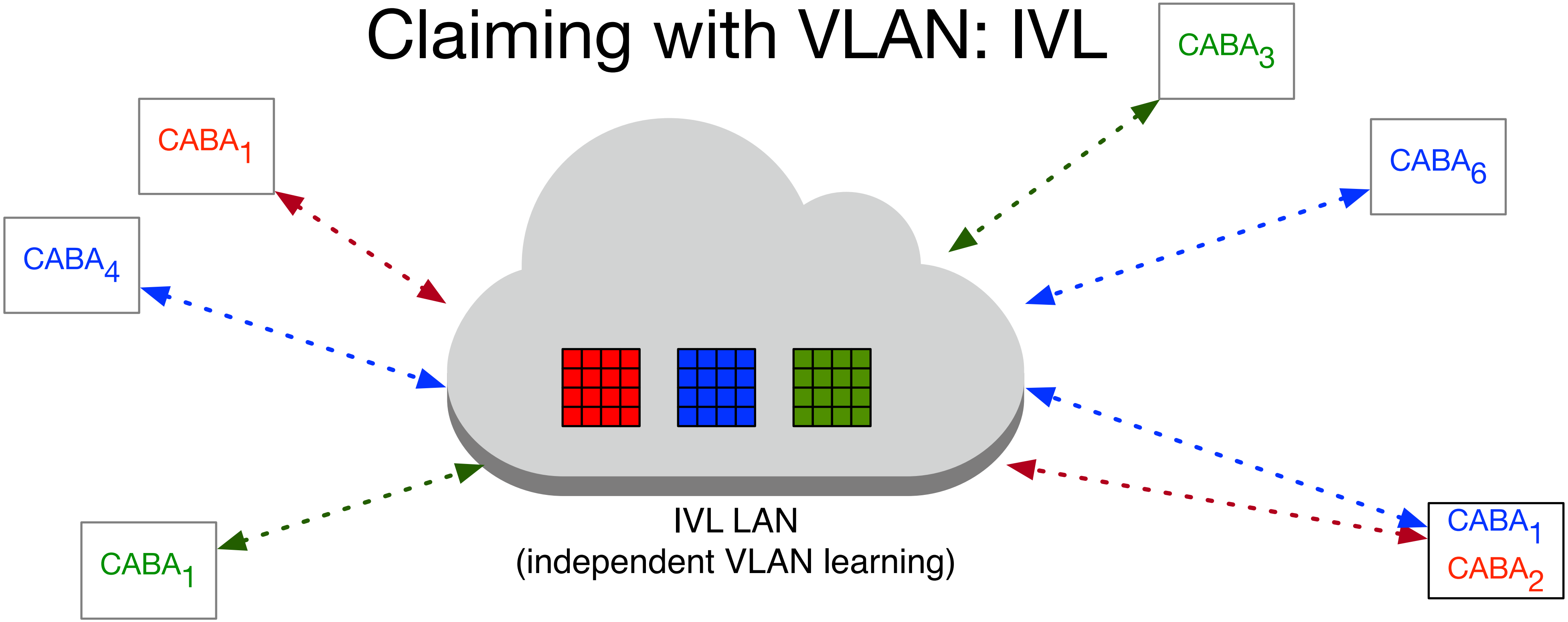
- Usage of any address may need to be limited to the VLAN or VLANs under which it was obtained.

- Due to the possibility that the same unicast address may be assigned in different VLANs, Independent VLAN Learning (IVL) may be needed in bridges, per IEEE Std 802.1Q Annex F (F.1.2).
  - This requirement could be relaxed in some cases
  - e.g. when assigned unicast addresses are declared via MMRP (instead of learning)

- This issue is common to claiming protocols generally.

- Some approaches follow.

Claiming with VLAN: IVL



IEEE Std 802-2014 says “Local MAC addresses need to be unique on a LAN or bridged LAN unless the bridges support VLANs with independent learning.”

With IVL, each VLAN has an independent forwarding table.

-but IVL is not always possible

BARC claiming on each VLAN is independent

a duplicate address may occur in more than one VLAN; that is not harmful if managed carefully

A claimant with multiple VLANs needs to claim in each VLAN.
Claimed address is usable only in claimed VLAN:

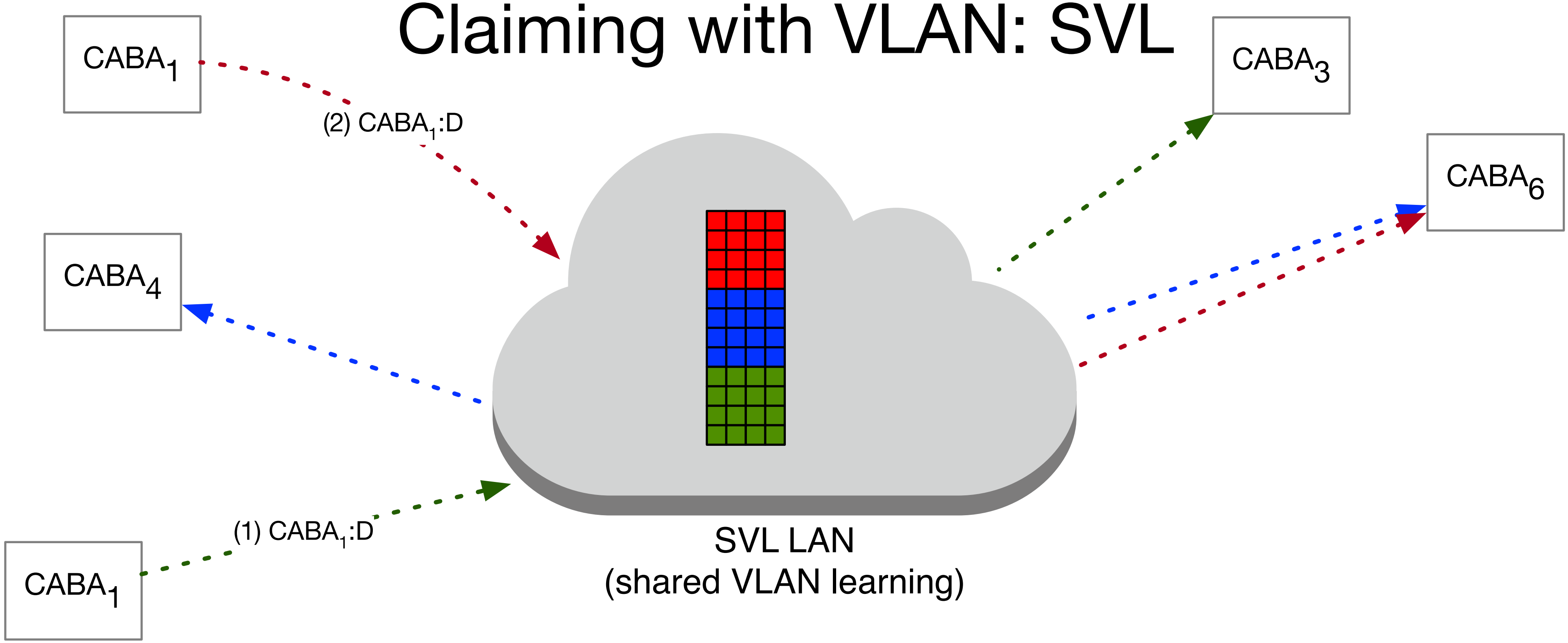
Claimant needs to bind address to VLAN

For convenience, Claimant may claim the same address in each of its VLANs

-Still, requires multiple claim messages and multiple forwarding table entries.

-Device needing many VLANs should consider an EUI

Claiming with VLAN: SVL



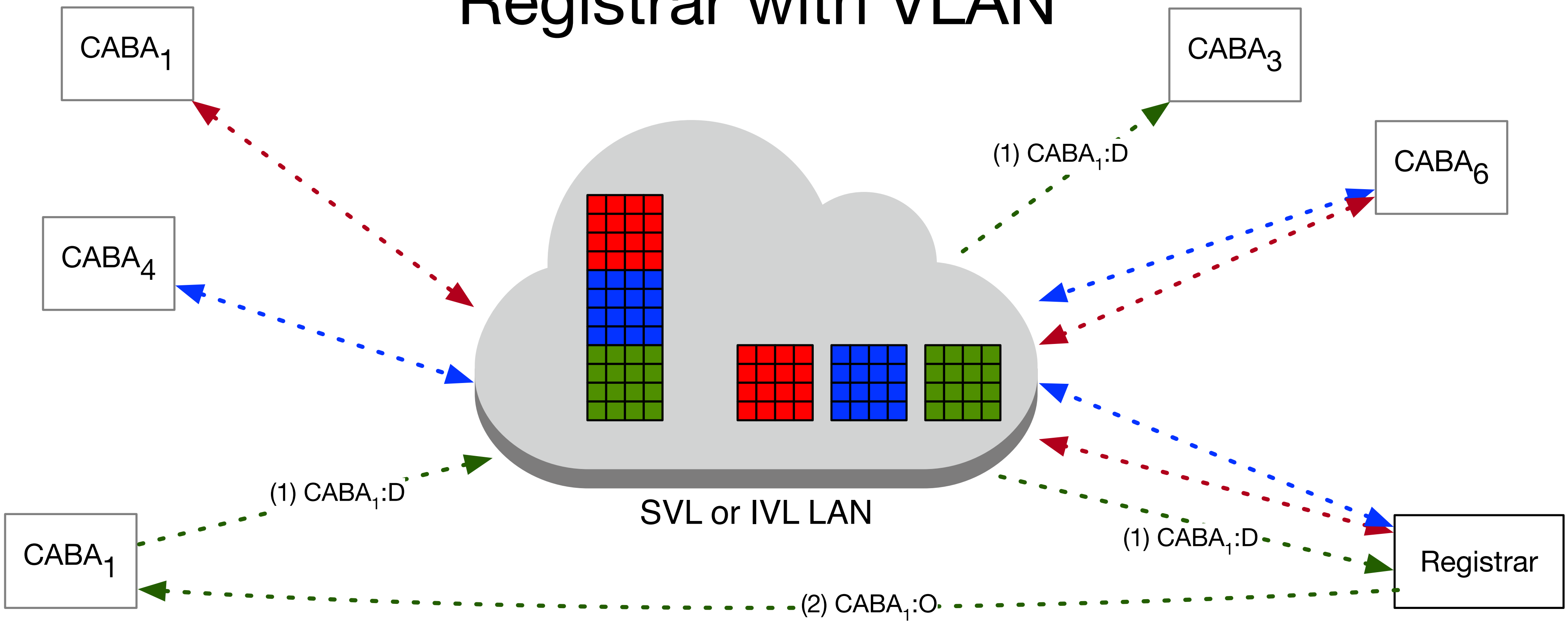
With SVL, VLANs share a forwarding table.

BARC claiming on each VLAN is independent

an address could become a duplicate, existing in more than one VLAN

forwarding table is limited to one entry per address, so duplication is catastrophic.

Registrar with VLAN



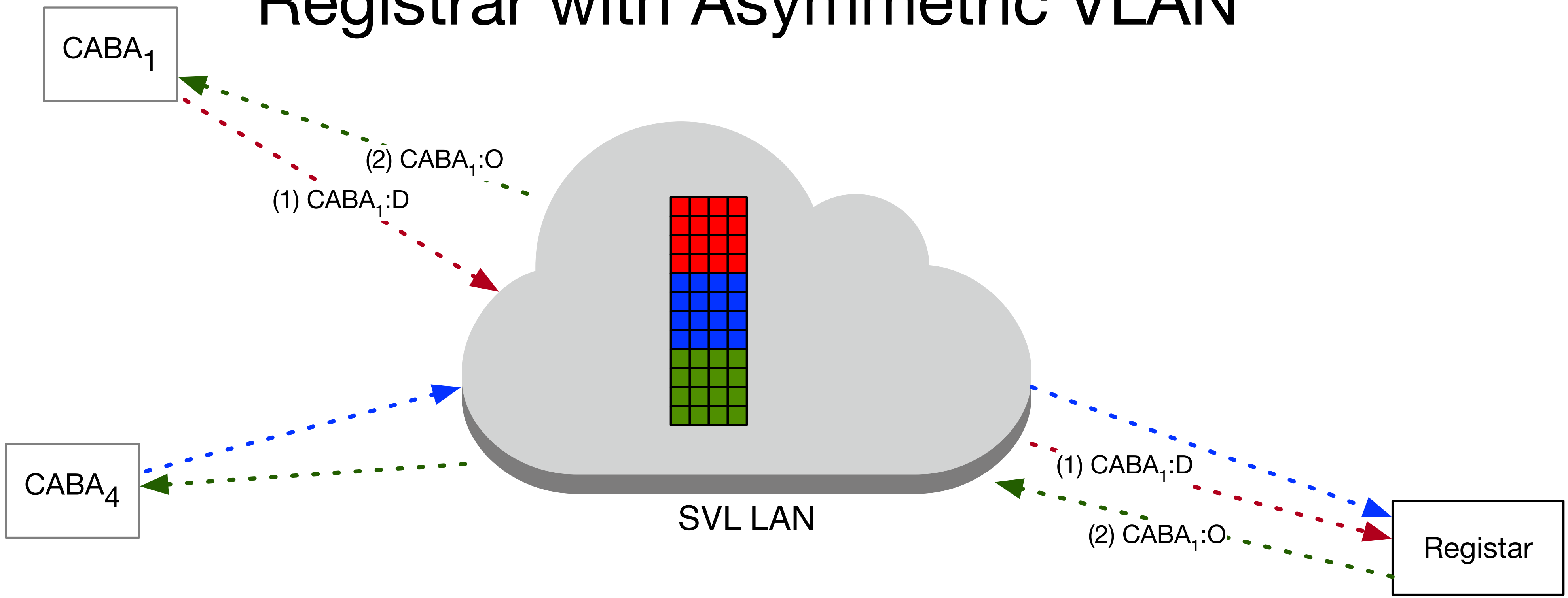
Network is configured with Registrar on all active VLANs on which BARC is used.

BARC claim from any VLAN is delivered to Registrar.
-Offer delivered on Claimant's VLAN

Registrar ensures that registered address is unique across all (or perhaps only some) of its VLANs.
-SVL or IVL will work

Registrar needs to remember over which VLANs the address was assigned.
-should be retained in State Machine

Registrar with Asymmetric VLAN



SVL is used for Asymmetric VLAN (IEEE Std 802.1Q Annex F.1.3)

Registrar can assign address to be unique across all VLANs available to the Registrar.

Summary

- Claimants operate with or without Registrars.

- Multiple registrars are supported, holding claims of disjoint RABIs.

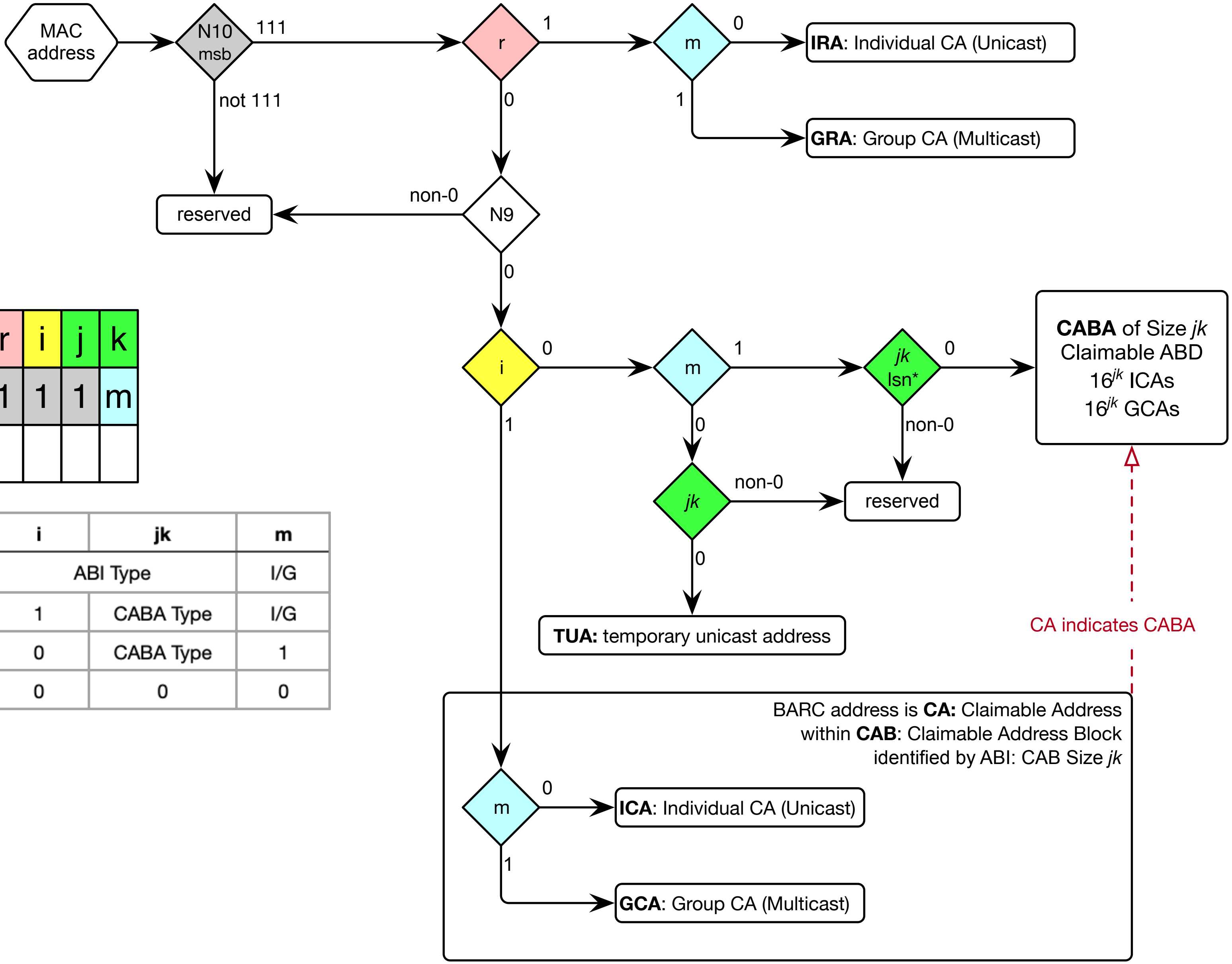
- The block discretization provides:
  - a vast set of addresses to a LAN
  - a large set of temporary unicast addresses
  - operational efficiency and simplicity
  - both unicast and multicast addresses to Claimant
    - unicast and multicast subblocks share the same range, except for the m bit
      - could be exploited
    - devices needing both unicast and multicast addresses need make only one claim

- Could integrate with MMRP to limit propagation and eliminate learning of unicast AB content.
  - MMRP needs to efficiently handle address ranges
  - BARP could be specified as alternative MRP application
    (e.g. would understand an ABD)

Appendix 1

- additional details on BARC addresses and identifiers

BARC Address Parsing



N11	r	i	j	k
N10	1	1	1	m
N9				

	r	i	jk	m
RA	1	ABI Type		I/G
CA	0	1	CABA Type	I/G
CABA	0	0	CABA Type	1
TUA	0	0	0	0

*lsn= least significant nibbles

CABA and CA, CAB Size 0-3

CAB Size C=0		CAB Size C=1		CAB Size C=2		CAB Size C=3	
CABA	CAB	CABA	CAB	CABA	CAB	CABA	CAB
0 0 0 0	0 1 0 0	0 0 0 1	0 1 0 1	0 0 1 0	0 1 1 0	0 0 1 1	0 1 1 1
1 1 1 1	1 1 1 *	1 1 1 1	1 1 1 *	1 1 1 1	1 1 1 *	1 1 1 1	1 1 1 *
0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
X8	X8	X8	X8	X8	X8	X8	X8
X7	X7	X7	X7	X7	X7	X7	X7
X6	X6	X6	X6	X6	X6	X6	X6
X5	X5	X5	X5	X5	X5	X5	X5
X4	X4	X4	X4	X4	X4	X4	X4
X3	X3	X3	X3	X3	X3	X3	X3
X2	X2	X2	X2	X2	X2	0	*
X1	X1	X1	X1	0	*	0	*
X0	X0	0	*	0	*	0	*

2 contiguous subblocks per CABA (one unicast, one multicast)

CAB Size C=0	CAB Size C=1	CAB Size C=2	CAB Size C=3
• 6.9E10 Size 0 CABAs • 1 CA/subblock	• 4.3E9 Size 1 CABAs • 16 CAs/subblock	• 2.7E8 Size 2 CABAs • 256 CA/subblock	• 1.7E7 Size 3 CABAs • 4096 CAs/subblock

* indicates wildcard (any value)

CABA/CAB Math

N11
N10

CABA

0	0	j	k
1	1	1	1

CA

0	1	j	k
1	1	1	*

•CABA indicates the CAB
•CA indicates the CABA
•*C*=*jk* indicates the CAB Size
•each CAB subblock includes 16^{C} contiguous addresses

&	bitwise AND
|	bitwise OR
~	bitwise NOT
/	divide

C(X) = (X&0x300000000000) / 0x100000000000 [extracts CAB Size *C* when X is CABA or CA]
Cmask(C) = ~(0x410000000000 + 0x10**C – 1) [CABA mask, per Size; used to create CABA from CA]
CABA(CA) = CA&Cmask(C(CA)) + 0x010000000000

Example: [Note: Underlining on the middle four nibbles is shown only as a reading aid.]

- CA = 0x6F0123456789 = 0110-1111-0000-0001-0010-0011-0100-0101-0110-0111-1000-1001
- C(CA) = 0x200000000000 / 0x100000000000 = 2
- Cmask(0x2) = ~ (0x410000000000 + 0x0100 - 1) = ~(0x4100000000FF) = 0xBEFFFFFFFF00
- CABA(CA) = CA&0xBEFFFFFFFF00 + 010000000000 = 0x2F0123456700

A CA is within CABA_x if and only if CABA(CA) = CABA_x

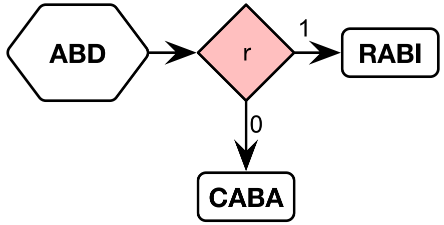
•this requires identical CAB Size

The CAB of CABA_x is the set of all CAs that satisfy CABA(CA) = CABA_x

Lowest ICA in CABA: ICAmin = (CABA-0x010000000000) | 0x400000000000 (example: 0x6E0123456700)
Highest ICA in CABA: ICAmax = ICAmin + 0x10**C(CABA) – 1 (example: 0x6E01234567FF)

Lowest GCA in CABA: GCAmin = CABA | 0x410000000000 (example: 0x6F0123456700)
Highest GCA in CABA: GCAmax = GCAmin + 0x10**C(CABA) – 1 (example: 0x6F01234567FF)

ABD (CABA/RABI) Format



	ABD: CABA				Address: CAB			
N11	0	0	j	k	0	1	j	k
N10	1	1	1	1	1	1	1	*

•CABA is both an ABD (indicating CAB) and a MAC address
•CAB indicates the CABA
•*jk* indicates the CABA size *C*
•the *C* least significant nibbles of the CABA are 0
•each CAB subblock includes 16^C contiguous addresses)
•each CAB includes a unicast subblock and a multicast subblock
•no CA within a CAB is within any other CAB (that is, a CAB with a different CABA

	ABD: RABI				Address: RAB			
N11	1	i	j	k	1	i	j	k
N10	a	b	c	d	1	1	1	*

•RABI is an ABD (indicating RAB) and never used as a MAC address
•RAB does not fully indicate the RABI
•*i* indicates RABI Option (Options 0 and 1 provide independent RABIs/RABs)
•*jk* indicates the BABI Size *B*
  - "BABI" for "Basic Address Block Indicator"
•*abcd* indicates MABI Size *M*; those bits are not in the RAB addresses
  - "MABI" for "Multiple Address Block Indicator"
•RAB Size $R=B+M$
•the *R* least significant nibbles of the RABI are 0
•each RAB subblock includes 16^R contiguous addresses
•each RAB includes a unicast subblock and a multicast subblock

RABI Aggregation

•RAB Size $R=B+M$
  • for 48-bit addresses, set $a=0$; then $R = jk + bcd \leq 3 + 7 = 10$, matching the 10 available nibbles N0-N9
  • could use *ijk* as the BABI Size, and/or the full *abcd* as the MABI Size; e.g., for 64-bit addresses
•A RABI may aggregate other RABIs.
•A RABI of RAB Size *R* and MABI Size *M* can be disaggregated into:
  •16 RABIs of RAB Size R–1 (MABI Size M–1), or
  •16^2 RABIs of RAB Size R–2 (MABI Size M–1), or
  •16^n RABIs of RAB Size R–n (MABI Size M–n), or
  • … 16^M RABIs of RAB Size *B* (MABI Size 0), or
•A RABI of RAB Size *B* (MABI Size 0) cannot be disaggregated.
•An RA appears in one and only RABI of each *M*.

RABI and RA, MABI Size 0, BABI Size 0-3

BABI Size 0		BABI Size 1		BABI Size 2		BABI Size 3	
RABI	RAB	RABI	RAB	RABI	RAB	RABI	RAB
1 i 0 0	1 i 0 0	1 i 0 1	1 i 0 1	1 i 1 0	1 i 1 0	1 i 1 1	1 i 1 1
0 0 0 0	1 1 1 *	0 0 0 0	1 1 1 *	0 0 0 0	1 1 1 *	0 0 0 0	1 1 1 *
X9	X9	X9	X9	X9	X9	X9	X9
X8	X8	X8	X8	X8	X8	X8	X8
X7	X7	X7	X7	X7	X7	X7	X7
X6	X6	X6	X6	X6	X6	X6	X6
X5	X5	X5	X5	X5	X5	X5	X5
X4	X4	X4	X4	X4	X4	X4	X4
X3	X3	X3	X3	X3	X3	X3	X3
X2	X2	X2	X2	X2	X2	0	*
X1	X1	X1	X1	0	*	0	*
X0	X0	0	*	0	*	0	*

2 contiguous subblocks per RABI (one unicast, one multicast)

- 1.1E12 Size 0 RABIs
- 1 RA/subblock

- 6.9E10 Size 1 RABIs
- 16 RAs/subblock

- 4.3E9 Size 2 RABIs
- 256 RA/subblock

- 2.7E8 Size 3 RABIs
- 4096 RAs/subblock

* indicates wildcard (any value)

Aggregation Example: BABI Size 3, various MABI Sizes

BABI Size 3									
MABI Size 0		MABI Size 1		MABI Size 2			MABI Size 7	
RABI	RAB	RABI	RAB	RABI	RAB		RABI	RAB
1 i 1 1	1 i 1 1	1 i 1 1	1 i 1 1	1 i 1 1	1 i 1 1		1 i 1 1	1 i 1 1
0 0 0 0	1 1 1 *	0 0 0 1	1 1 1 *	0 0 1 0	1 1 1 *		0 1 1 1	1 1 1 *
X9	X9	X9	X9	X9	X9		0	#
X8	X8	X8	X8	X8	X8		0	#
X7	X7	X7	X7	X7	X7		0	#
X6	X6	X6	X6	X6	X6	…	0	#
X5	X5	X5	X5	X5	X5		0	#
X4	X4	X4	X4	0	#		0	#
X3	X3	0	#	0	#		0	#
0	*	0	*	0	*		0	*
0	*	0	*	0	*		0	*
0	*	0	*	0	*		0	*

aggregate

aggregate

aggregate

* indicates wildcard (any value) # indicates wildcard (any value)

Example:
Hierarchical RABI Addressing with common Registrar

Held by Registrar		Assigned to Bridge 1 by Registrar		Assigned by Registrar to bridges, end stations, etc. connected to Bridge 1			
MABI Size 2		MABI Size 1		MABI Size 1		MABI Size 1	
RABI	RAB	RABI	RAB	RABI	RAB	RABI	RAB
1 i 1 1	1 i 1 1	1 i 1 1	1 i 1 1	1 i 1 1	1 i 1 1	1 i 1 1	1 i 1 1
0 0 1 0	1 1 1 *	0 0 0 1	1 1 1 *	0 0 0 1	1 1 1 *	0 0 0 1	1 1 1 *
X9	X9	X9	X9	X9	X9	X9	X9
X8	X8	X8	X8	X8	X8	X8	X8
X7	X7	X7	X7	X7	X7	X7	X7
X6	X6	X6	X6	X6	X6	X6	X6
X5	X5	X5	X5	X5	X5	X5	X5
0	#	8-15	8-15	0	0	1	1
0	#	0	#	0	#	0	#
0	*	0	*	0	*	0	*
0	*	0	*	0	*	0	*
0	*	0	*	0	*	0	*

common RAB prefix

…

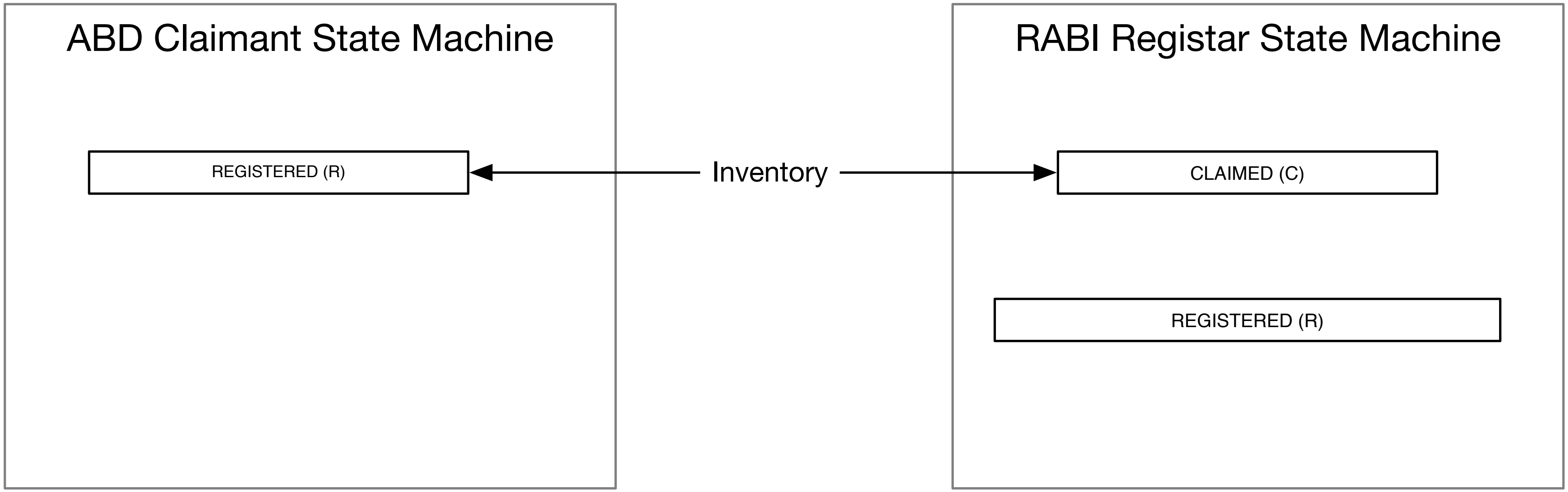
Bridge serves as Advisor to connected station; proposes PRABI, and Registrar Address, in response to an Inquiry.

Example:
Hierarchical CABA Addressing

Claimed by Bridge 1			Claimed by bridges, end stations, etc. connected to Bridge 1			
CAB Size C=3			CAB Size C=2		CAB Size C=2	
CABA	CAB		CABA	CAB	CABA	CAB
0 0 1 1	0 1 1 1		0 0 1 0	0 1 1 0	0 0 1 0	0 1 1 0
1 1 1 1	1 1 1 *	common CAB prefix	1 1 1 1	1 1 1 *	1 1 1 1	1 1 1 *
0 0 0 0	0 0 0 0		0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
X8	X8		X8	X8	X8	X8
X7	X7		X7	X7	X7	X7
X6	X6		X6	X6	X6	X6
X5	X5		X5	X5	X5	X5
X4	X4		X4	X4	X4	X4
X3	X3		X3	X3	X3	X3
0	*		0	0	1	1
0	*		0	*	0	*
0	*		0	*	0	*

Bridge serves as Advisor to connected stations; proposes CABA in response to an Inquiry.

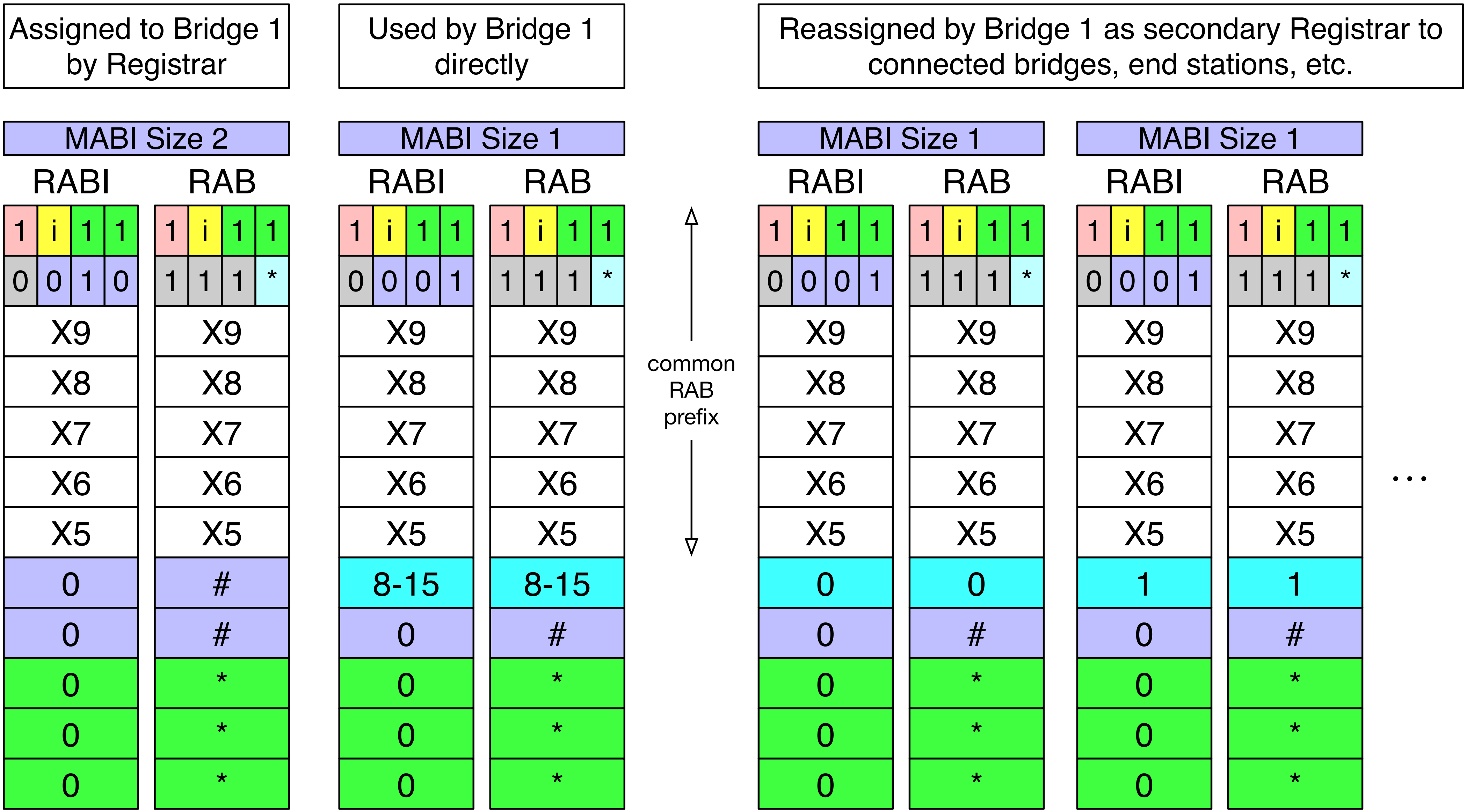
Claimant as Registrar



RABI registered as a Claimant could be disaggregated and reassigned by a Registrar function managed jointly with the Claimant.

A RABI in the “REGISTERED” state of the ABD Claimant State Machine could be considered to be in the Inventory of the RABI Registrar State Machine (along with Claimed RABIs) and could be disaggregated, Offered and Registered by that Registrar.

Example: Hierarchical RABI Disaggregation with Tiered Registrars



RABI Math 101

N11
N10

RABI

1	i	j	k
0	b	c	d

RA

1	i	j	k
1	1	1	*

•*jk* indicates the BABI Size *B*
•*bcd* indicates the MABI Size *M*

&	bitwise AND
~	bitwise NOT
/	divide
==	binary equality

i(X) = (X&0x400000000000) / 0x400000000000 [RABI Option, when X is RABI or RA]
B(X) = (X&0x300000000000) / 0x100000000000 [BABI Size, when X is RABI or RA]
M(X) = (X&0x070000000000) / 0x010000000000 [MABI Size when X is a RABI]
R(X) = B(X) + M(X) [RAB Size, when X is a RABI]
Rmask(N) = ~(0x0F0000000000 + 0x10**N – 1) [mask, used below]

An RA is within RABIx if and only if RA&Rmask(R(RABIx)) = RABIx&Rmask(R(RABIx))
•this requires identical RABI Option and BABI Size
The RAB of RABIx is the set of all RAs that satisfy RA&Rmask(R(RABIx)) = RABIx&Rmask(R(RABIx))

Example:
- RABI = 0xF20123400000 = 1111-0010-0000-0001-0010-0011-0100-0000-0000-0000-0000-0000
- B(RABI) = 0x300000000000 / 0x100000000000 = 3
- M(RABI) = 0x020000000000 / 0x010000000000 = 2
- R(RABI) = 3+2 = 5
- Rmask(5) = ~(0x0F0000000000 + 0x10**5 – 1) = ~(0x0F00000FFFFF)= 0xF0FFFFF00000
- RABI&Rmask(5) = 0xF00123400000

- RA = 0xFE0123456789 = 1111-1110-0000-0001-0010-0011-0100-0101-0110-0111-1000-1001
- RA&Rmask(5) = 0xF00123400000 = RABI&Rmask(5)
- so RA is within RABI

RABI Math 102

N11
N10

RABI

1	i	j	k
0	b	c	d

RA

1	i	j	k
1	1	1	*

•*jk* indicates the BABI Size *B*
•*bcd* indicates the MABI Size *M*

&	bitwise AND
~	bitwise NOT
/	divide
==	binary equality

RABIcheck(RABI1,RABI2) determines whether RABI RABI1 overlaps RABI2.
RABIcheck(R1,R2) = R1&Rmask(R(R2))
RABI1 shares RAs with RABI2 if and only if:
RABIcheck(RABI1, RABI2) = RABIcheck(RABI2, RABI1)
•Note: This can be ruled out by inspection if the two N11 nibbles differ

Example:

- RABI2 = `0xF50100000000` = `1111-0101-0000-0001-0000-0000-0000-0000-0000-0000-0000-0000`
- B(RABI2) = `0x300000000000` / `0x100000000000` = 3
- M(RABI2) = `0x070000000000` / `0x100000000000` = 5
- R(RABI2) = 3+5 = 8
- Rmask(8) = `0xF0FF00000000`
- RABI1 = `0xF20123400000`
- RABI1&Rmask(8) = `0xF00100000000`
- RABI2&Rmask(5) = `0xF00100000000`
- so RABI1 and RABI2 have RAs in common

Lowest IRA in RABI: IRAmin = RABI & `0xF0FFFFFFFFFF + 0x0E0000000000`
Example: `0xFE0123400000`)
Highest IRA in RABI: IRAmax = IRAmin + `0x10`**(R(RABI)) – 1
Example: `0xFE01234FFFFF`)
Lowest GRA in RABI: GRAmin = RABI & `0xF0FFFFFFFFFF + 0x0F0000000000`
Example: `0xFF0123400000`)
Highest GRA in RABI: GRAmax = GRAmin + `0x10`**(R(RABI)) – 1
Example: `0xFF01234FFFFF)`

RABI Math 103

N11
N10

RABI

1	i	j	k
0	b	c	d

RA

1	i	j	k
1	1	1	*

•*jk* indicates the BABI Size *B*
•*bcd* indicates the MABI Size *M*

&	bitwise AND
~	bitwise NOT
/	divide
==	binary equality

An RA exists in one and only one RABI (called RABI_W) of each MABI Size *W*.
Given the MABI Size *W*, what is RABI_W?

A RABI exists in one and only one aggregated RABI called RABI) of each larger MABI Size *W*.
Given the MABI Size *W*, what is RABI_W?

$\text{RABI}_W(X, W)$ = X&Rmask(*W*+B(X)) + *W**`0x010000000000` [X is an RA, or a RABI with M(X)<*W*]

Example:

- RABI1 = `0xF20123400000` B(RABI1)=3; M(RABI1)=2
- *W* = 5; *W* + B = 8
- RABI1&Rmask(8) = `0xF20123400000&0xF0FF00000000` = `0xF00100000000`
- RABIm(RABI1,5) = `0xF00100000000 + 0x050000000000 = 0xF50100000000`

Example:

- RA = `0xFE0123456789 [B(RA)=3]`
- *W* = 5; *W* + B = 8
- RA&Rmask(8) = `0xFE0123456789&0xF0FF00000000` = `0xF00100000000`
- RABIm(RA,8) = `0xF00100000000 + 0x050000000000 = 0xF50100000000`

Example:

- RA = `0xFE0123456789 [B(RA)=3]`
- *W* = 2; *W* + B = 5
- RA&Rmask(5) = `0xFE0123456789&0xF0FFFF00000` = `0xF00123400000`
- RABIm(RA,5) = `0xF00123400000 + 0x020000000000 = 0xF20123400000`

Temporary Unicast Address (TUA)

- For temporary use

- device without a source address selects a random temporary unicast address for initial discovery <u>only</u>
  - protocol then assigns at least one persistent unicast address

- simultaneous duplicate temporary addresses may lead to message loss in some circumstances
  - network learns route to source as initial message crosses the network
  - before response is returned, another initial message with duplicate source address crosses the path and rewrites the route
  - unlikely to be disastrous
  - loss of initial message will be corrected eventually

- nevertheless, need to consider the likelihood of duplication

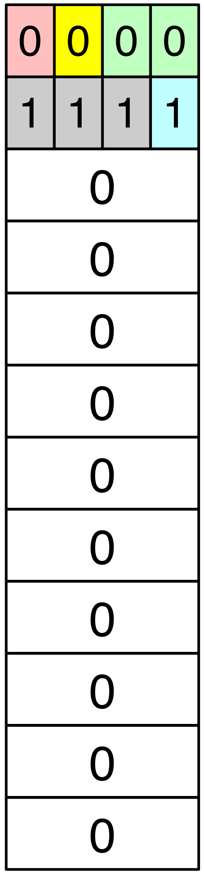
- Temporary address range includes 9 full nibbles of 16 values each (0–F)
  - 16^9 = 68,719,476,736 (= N) temporary addresses in the pool
  - chance of no duplicates with k randomly selected addresses is approximated $\exp(-k*(k-1)/(2*N))$
  - with k=1000 devices <u>simultaneously</u> using a temporary address, chance of no duplicates is ~0.99988
  - can add jk bits to the pool if more entropy is needed

0	0	0	0
1	1	1	0
0	0	0	0
*			
*			
*			
*			
*			
*			
*			
*			
*			

null CABA ($CABA_0$)

- Null CABA ($CABA_0$) is not an assignable address.
- Registrar listens to $CABA_0$.
- No Claimant listens to $CABA_0$.
- Can be used as the DA of BARC Inquiry; e.g., when Registrar address is unknown.

$CABA_0$



Proposed RABI (PRABI) and null RABI/PRABI

- Usable only as the content of a BARC Inquiry or BARC Proposal message.
- Indicates a set of RABIs characterized by RABI Option *i*, BABI Size B *jk*, and MABI Size M *bcd*.
- If the PRABI has the form of a RABI (with 0 in the B+M least significant nibbles), then the PRABI set is only that RABI.
- Non-zero values in the lower B+M nibbles can signify that some bits of the higher nibbles are “don’t care.”
- For example, the lower B+B nibbles of the PRABI could form a bitmask of the higher nibbles.

PRABI

1	i	1	1
0	0	1	0

X9
X8
X7
X6
X5
F
0
F
0
0

proposed RABIs

1	i	1	1
1	1	1	*

	don’t care:
~~X9~~	F
X8	0
~~X7~~	F
X6	0
X5	0
0	
0	
0	
0	
0	

- As a PRABI in a BARC Inquiry or Proposal message, indicates a set of RABIs characterized by RABI Option *i*, BABI Size B *jk*, and MABI Size M *bcd*, without expressing a preference for the RABI values of the 0 nibbles.
- As a RABI in a BARC Offer message, indicates to Claimant that no RABI is offered.

null RABI/ null PRABI

1	i	1	1
0	b	c	c

0
0
0
0
0
0
0
0
0
0

Proposed RABIs are those that satisfy:
RABI | Pmask = (PRABI&~(0x10**(R(PRABI)) – 1)) | Pmask
where Pmask(PRABI) = 10**(R(PRABI))*PRABI&(0x10**(Rcap) – 1)
and Rcap=min(R(PRABI),10–R(PRABI))

Appendix 2

- additional procedural details

BARC Address Propagation with MMRP

The ARC Claimant Application AddABD Process includes “declare with MMRP”. This entails declaring, to MMRP (when available), MMRP attributes, using an MMRPDU per IEEE Std 802.1Q § 10.12.1.6:

- The multicast address represented by the CABA
  - FirstValue field = CABA/NumberOfValues=1
- The unicast address subblock indicated by the ABD (CABA or RABI)
  - FirstValue field = first address in unicast subblock
  - NumberOfValues = 16^{Size} per ABD Size
    - limited to Size up to 3: 16^3=4096, and MRP provides 13 bits of NumberOfValues (2^{13}=8192)

The ARC Claimant Application AddABD Process includes (“select CABA”); this selection should consider any local MMRP registration database to avoid selecting a registered CABA.

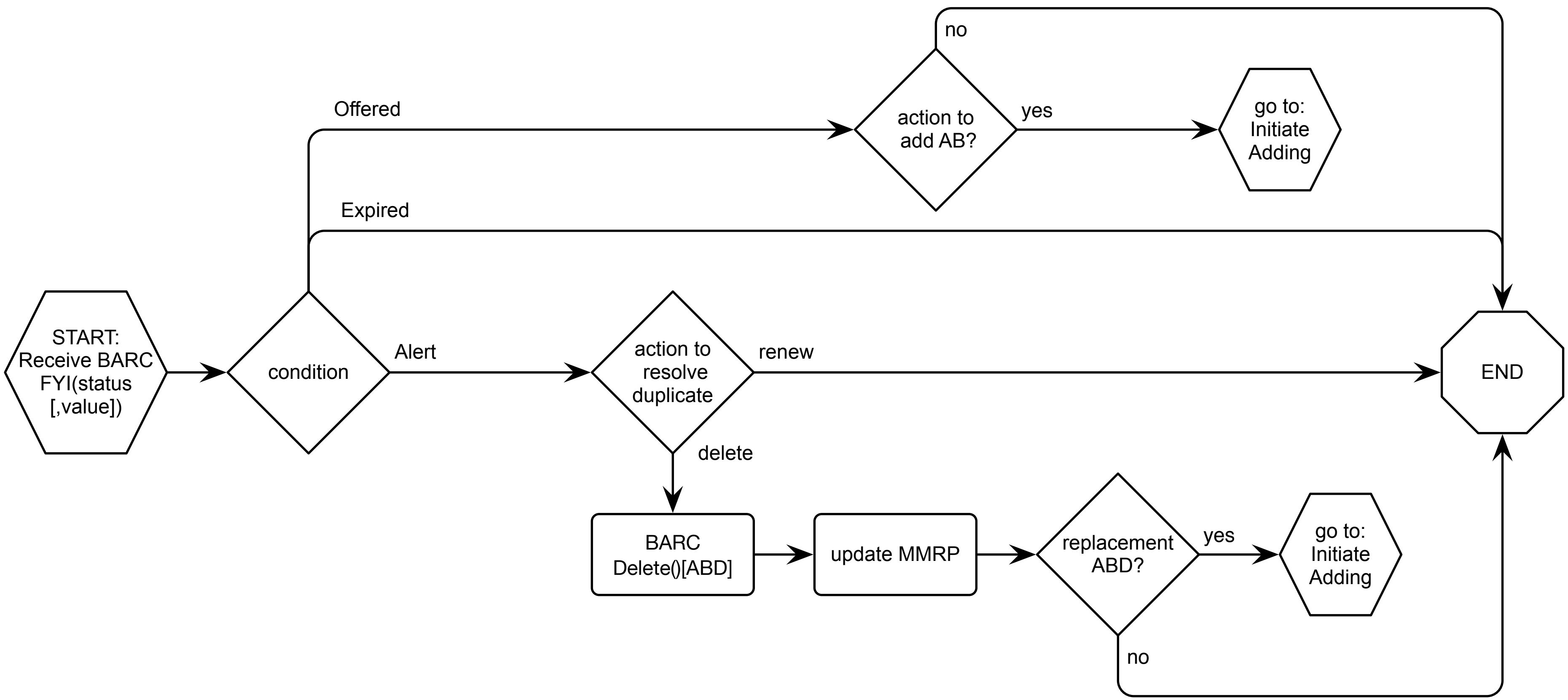
Unicast MMRP declaration can be useful because:
(1) A one-step declaration covers a contiguous range of self-assigned unicast addresses.
(2) Eliminates flooding for all the unicast addresses in the assignment.
(3) Eliminates the need for learning of each unicast address when used.
(4) Precludes erroneous re-learning of an address when a false duplicate is used elsewhere in the network.
- Could be a way to control duplication.
- Security issues to study.

BARC could alternatively specify “BARP,” a new MRP application. This could entail the following changes:

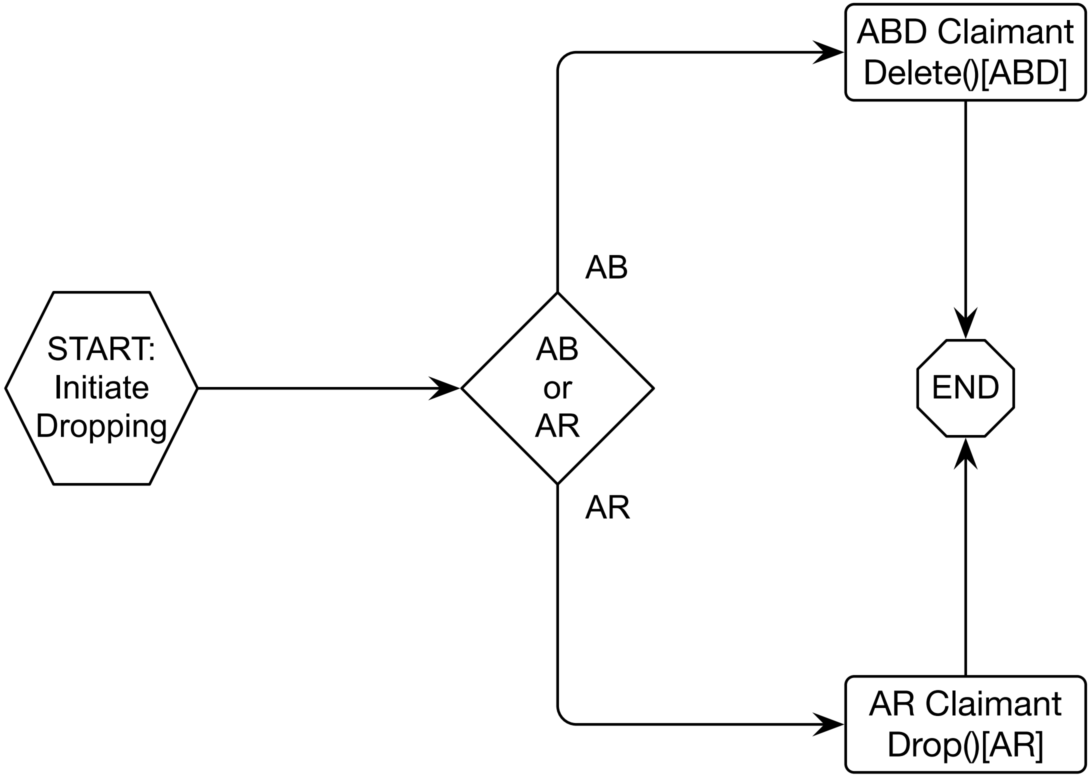
(a) the BARP application would be enabled to Join and Leave with the ABD as the declared attribute
(b) the BARP application would be specified to understand the semantics of the ABD and extract from it the indicated ABD multicast address and the indicated unicast address set, then use it to populate the FDB
(c) In the BARC Claimant ABD State Machine, the CABA claim [“sBARC(CABA:C)]” might not be needed, since the a BARP declaration could convey the claim to the CABA as well as the declaration of interest in receiving at the CABA multicast address

BARP might be better suited to specification within IEEE Std 802.1Q instead of 802.1CQ.

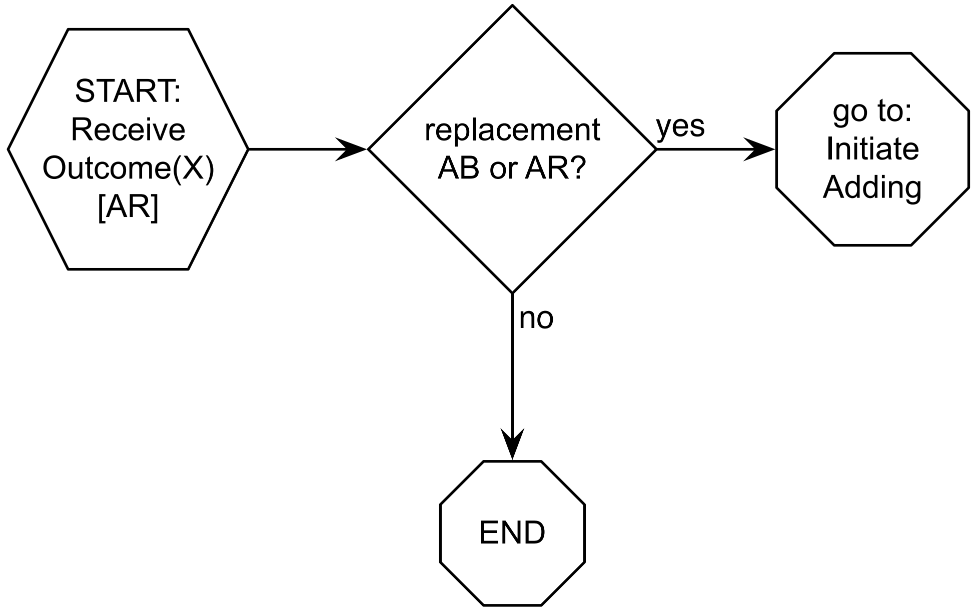
ARC Claimant Application Process: BARC Management



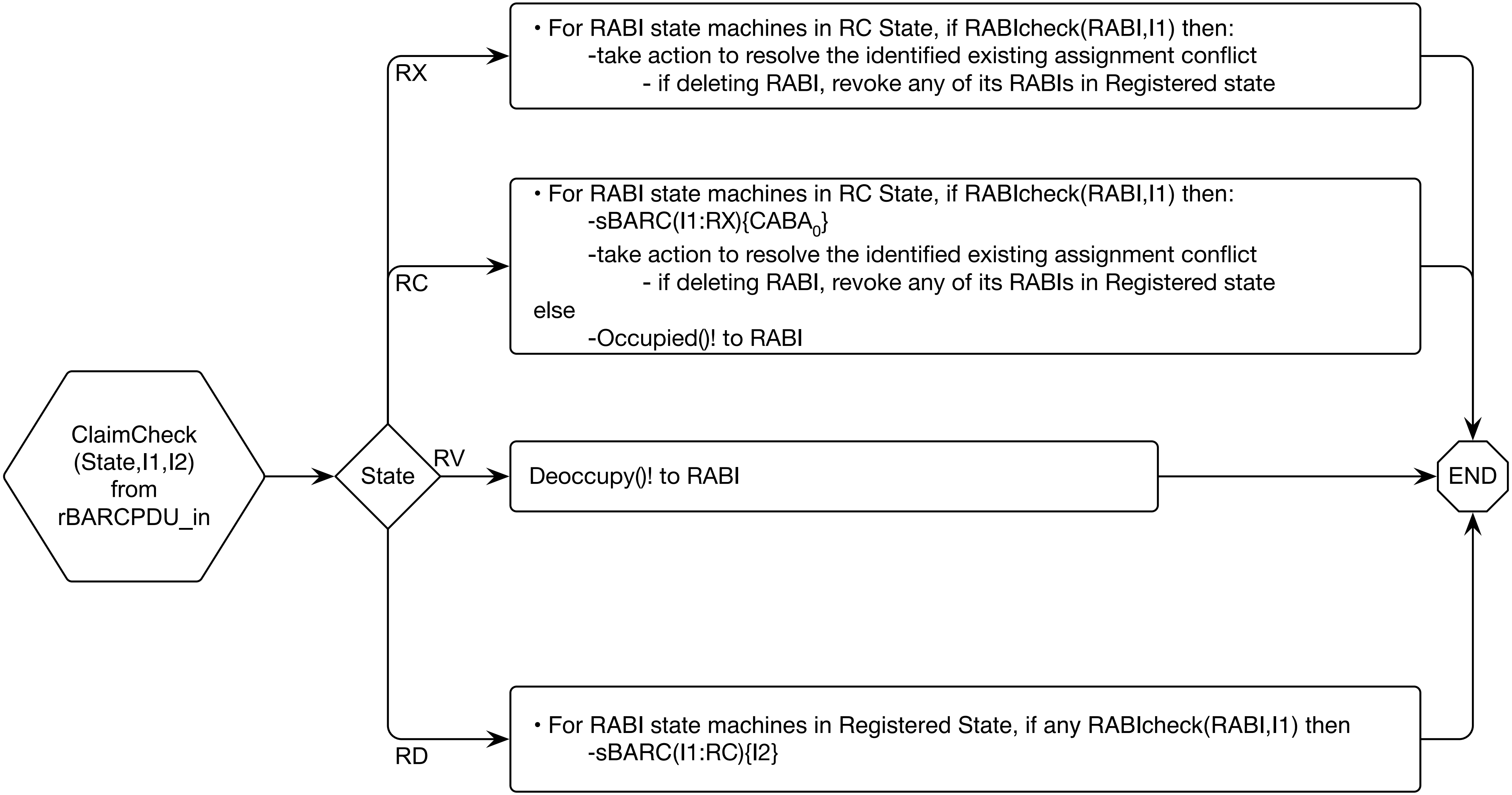
ARC Claimant Application Process: Drop Claim



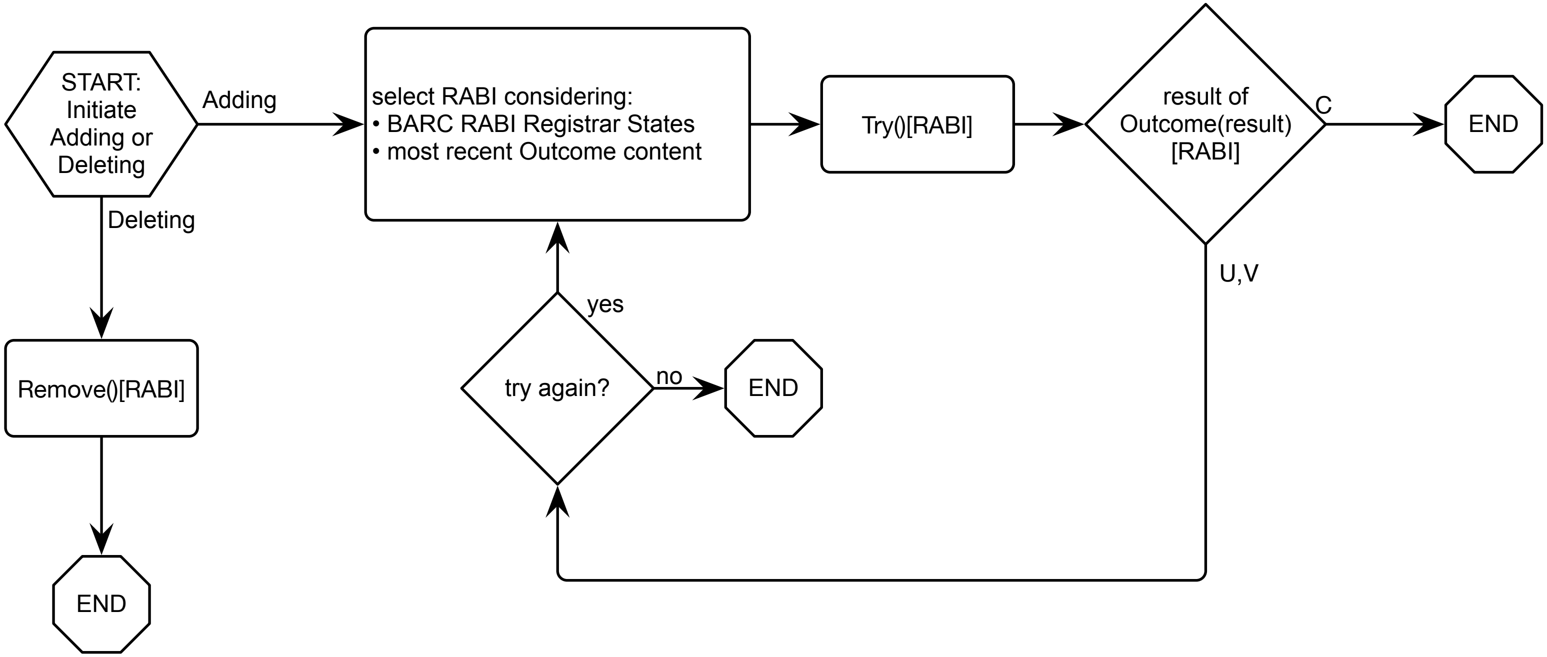
ARC Claimant Application Process: MAAP Management



BARC Registrar Application: ClaimCheck Process



BARC Registrar Application: RABI Claiming



RABI Claimant Application Process: BARC Management