P802.1CQ
Block Claims

Roger Marks
EthAirNet Associates
2020-10-26

Issues with Address Claiming

Claiming, per P802.1CQ/D0.5:

(1) Devices are frequently interrupted by messages to the claiming multicast address
- –repeated DISCOVER/PROBE messages
- –periodic ANNOUNCE messages
- –devices need need to receive and process each of these messages
  - –extract the claim and respond with DEFEND if necessary

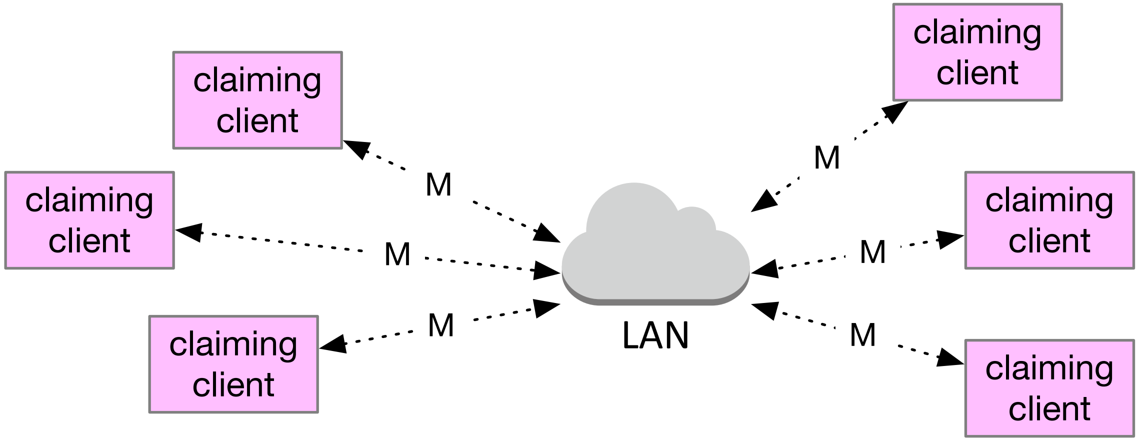
(2) Network forwards claiming multicast messages to all claiming devices

(3) Disallows a claim for both unicast and multicast addresses
- –does not support need for a multicast claim by a device without a source address
- –requires two claims (slow and chatty)
- –could change to support a simultaneous claim for unicast and multicast
  - –adds complexity to examination for conflicts; may result in more DEFENDs

It’s possible to address these issues with block claims, as proposed here.

These issues exist with PALMA (as of P802.1CQ/D0.5) and MAAPv1 as well.

Claim Messages



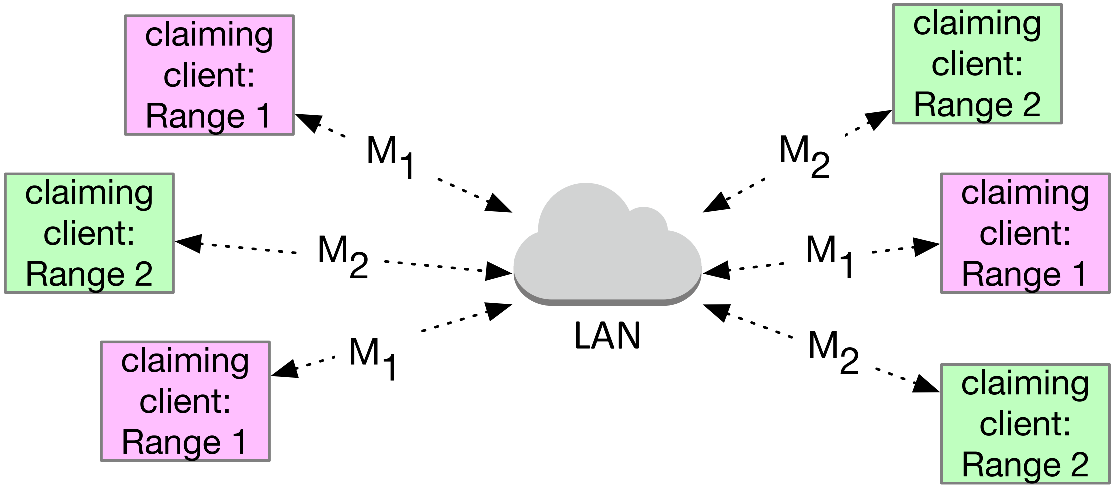
Per P802.1CQ/D0.5, N clients send claim messages to claiming multicast address M

- DISCOVER/PROBE messages at start
- ANNOUNCE messages periodically state the claimed address set
- Each message is received by (N–1) claiming devices listening to the claiming multicast address M
- Each message is processed by the claiming client
- N^2 messages delivered and processed, periodically

Possible improvement

- The purpose of DISCOVER/PROBE and ANNOUNCE is to identify address conflicts.
- Per MAAP specification, "All MAAP_PROBE, MAAP_DEFEND, and MAAP_ANNOUNCE PDUs that do not conflict with the address range associated with this state machine are ignored."
  - However, they must be delivered, received, and analyzed in order to confirm the lack of conflict; only then can they be ignored.
- Can we avoid reception of the message when no conflict is possible?

Divided Multicast Range: Elementary Example



Example: Half of claiming clients claim addresses in Range 1; the other half claim addresses in Range 2.

- Range 1 and Range 2 are disjoint, so claims don’t conflict.
- Use separate multicast addresses: M_1 for Range 1 clients; M_2 for Range 2 clients.

Could arrange for N/2 Range 1 clients to periodically send ANNOUNCE message to claiming multicast address M_1

- Each is received by (N/2–1) claiming Range 1 devices listening to the claiming multicast address M_1
- Each message is processed by the claiming Range 1 client (extract address set and compare for duplicates)
- $(N/2)^2$ messages delivered and processed, periodically

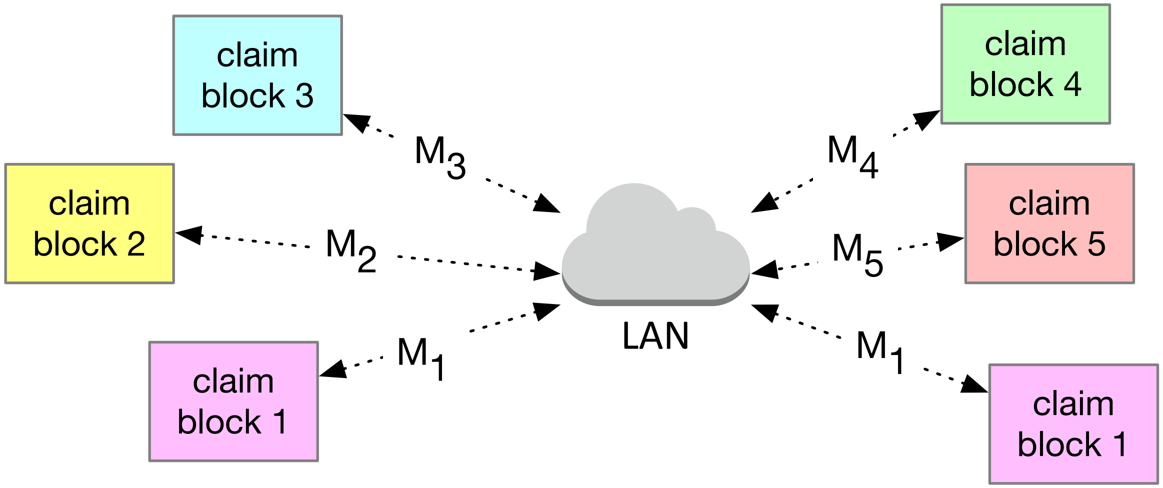
Same for Range 2 clients

=> $N^2/2$ messages delivered and processed, periodically

- half as many interruptions per device

But we can subdivide further, since there is no shortage of multicast addresses.

Claim Blocks



- Specify discrete and disjoint address claim blocks.

- Associate each address claim block with an identifier in the form of a multicast address.

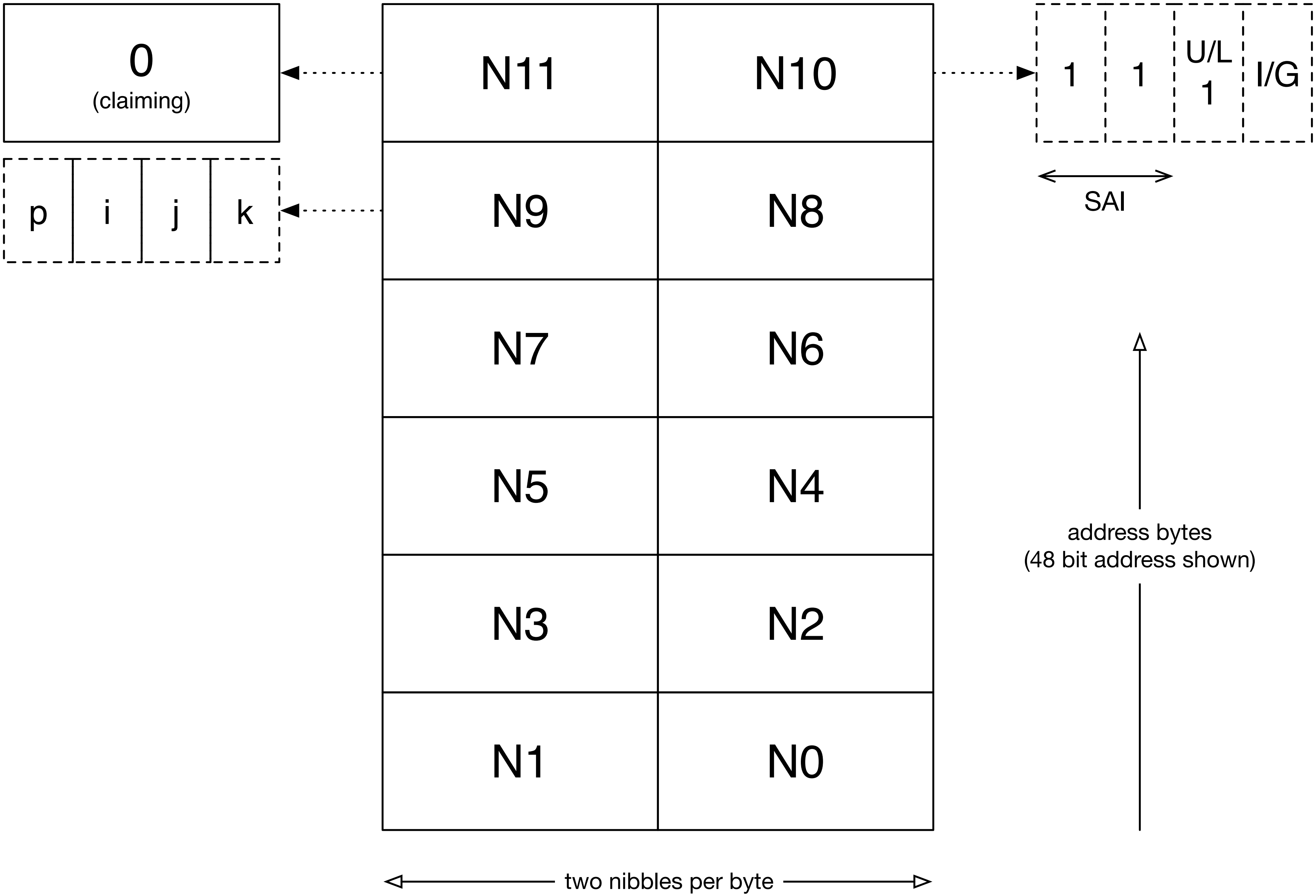
- Device claims a block by reference to the identifier, in the PDU, sent to that multicast address.

- Claiming client listens for claiming multicast only at the multicast identifier of its own claim block.
  - multicast claiming message will be received only by devices sharing the same claim block
  - for operational purposes, other client devices don't need the information
    - if they want to know, for some other reason, they can listen at all claiming addresses
    - address server listens to all so it can hear DISCOVER/ANNOUNCE

- If bridges support Extended Filtering Services, use Multiple MAC Registration Protocol (MMRP, 802.1Q §10.9–12), so that multicast claims will be delivered only to devices with matching claim block.

- How to specify and identify claim blocks?

802.1CQ Address Structure



0
(claiming)

p | i | j | k

N11	N10
N9	N8
N7	N6
N5	N4
N3	N2
N1	N0

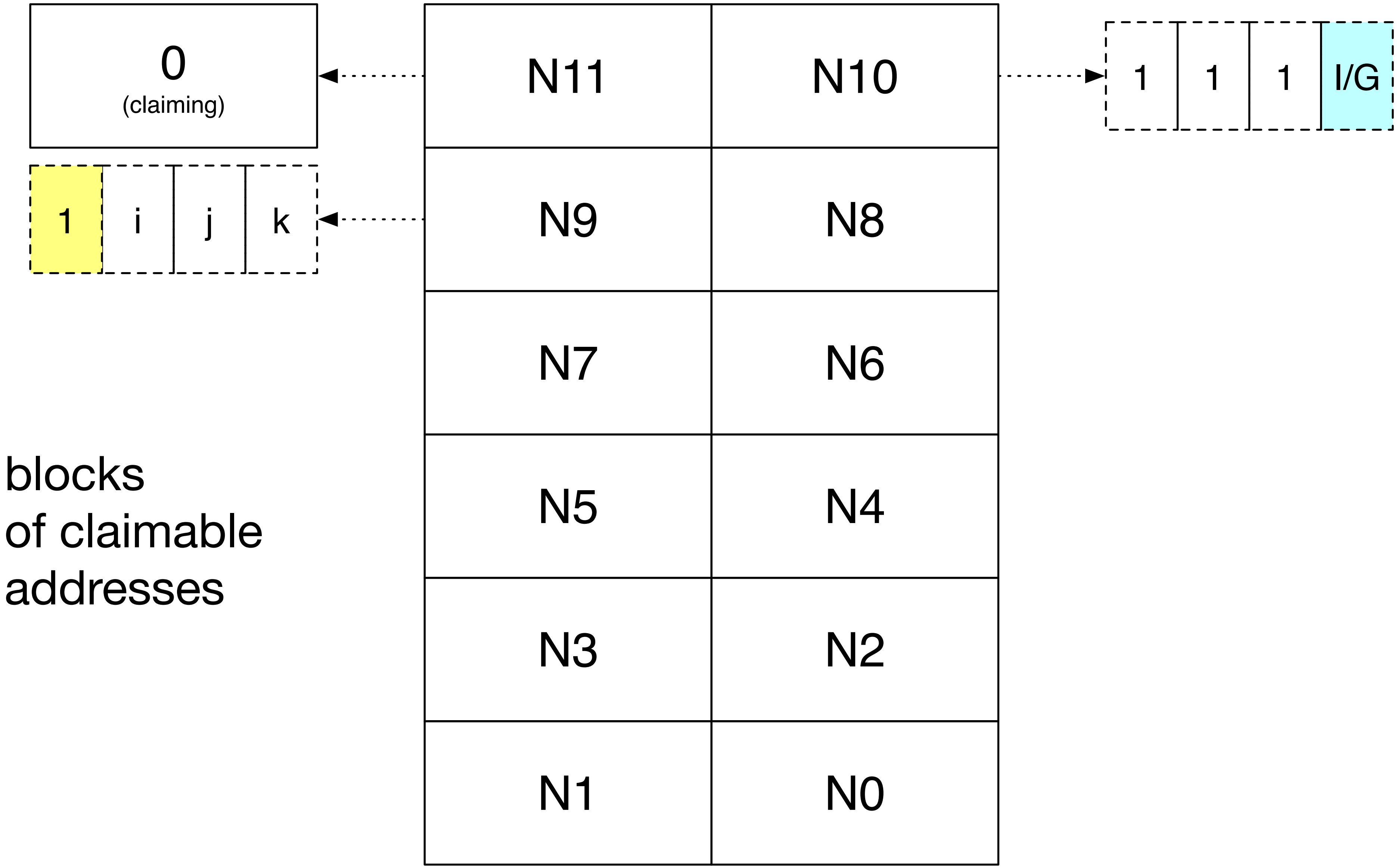
1	1	U/L 1	I/G

SAI

address bytes
(48 bit address shown)

two nibbles per byte

Claim Blocks



0
(claiming)

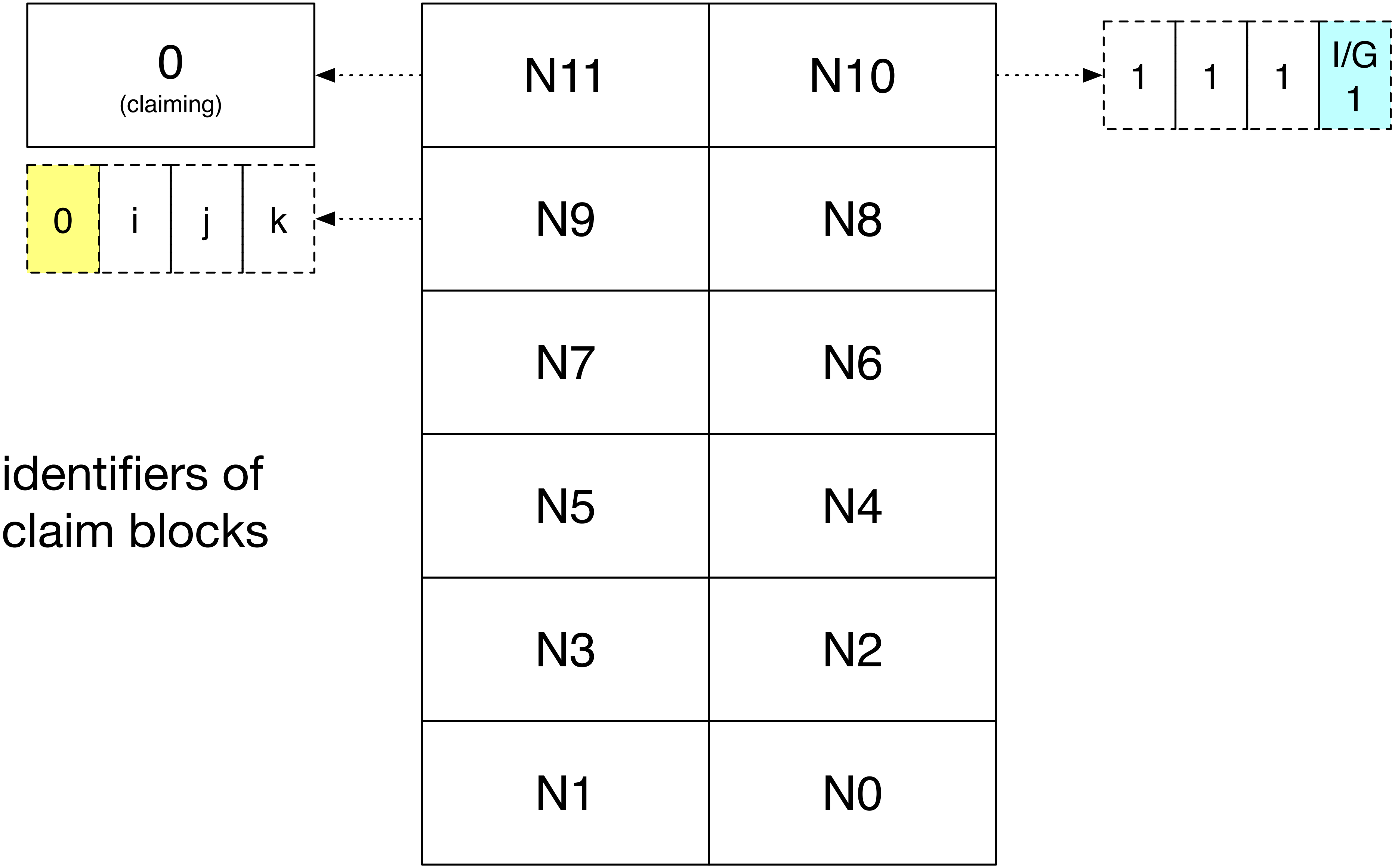
1	i	j	k

N11	N10
N9	N8
N7	N6
N5	N4
N3	N2
N1	N0

1	1	1	I/G

blocks
of claimable
addresses

Claim Block Identifiers



0
(claiming)

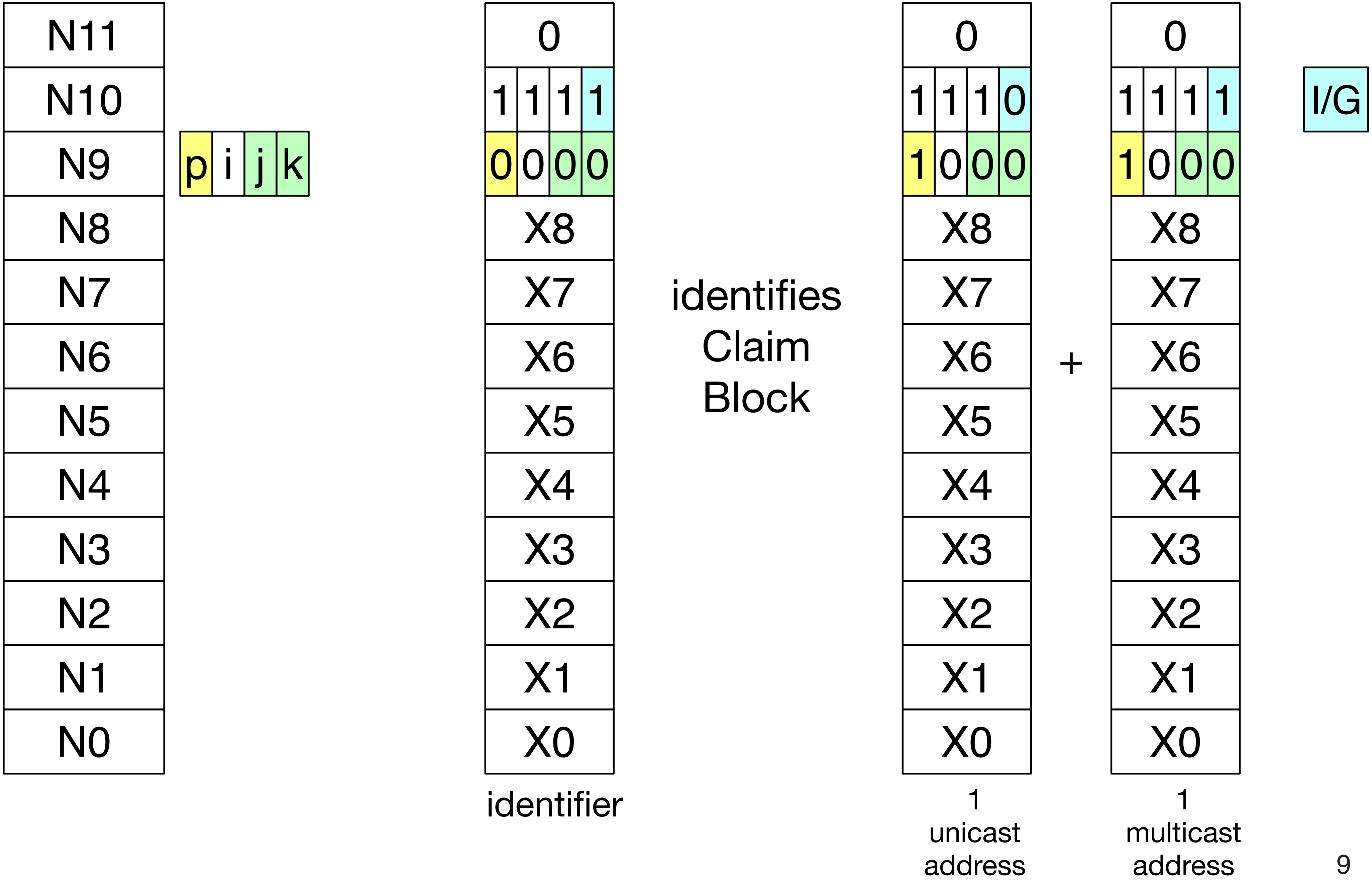
0	i	j	k

N11	N10
N9	N8
N7	N6
N5	N4
N3	N2
N1	N0

1	1	1	I/G 1

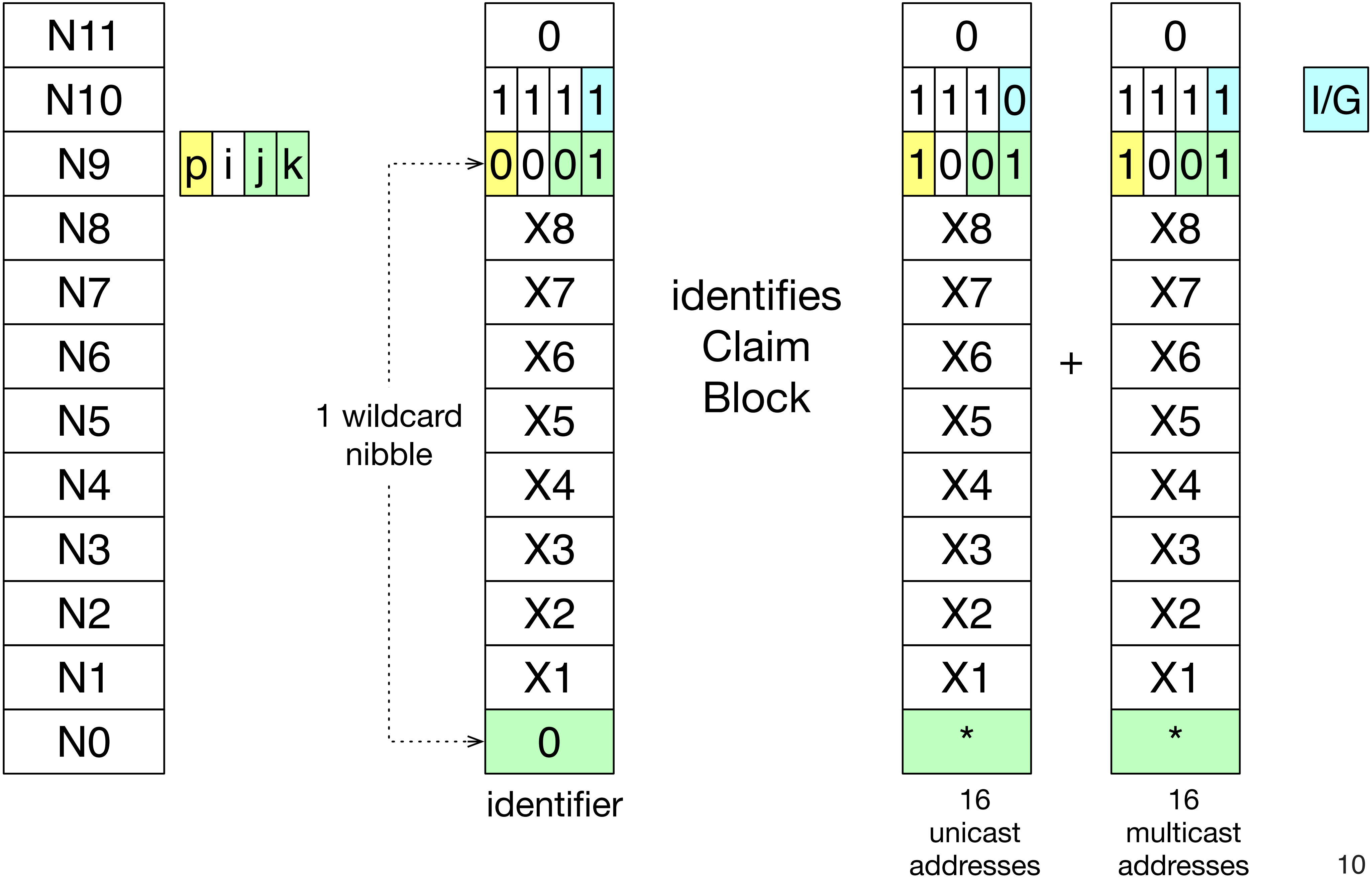
identifiers of
claim blocks

Claim Blocks and Identifiers (*jk*=00)

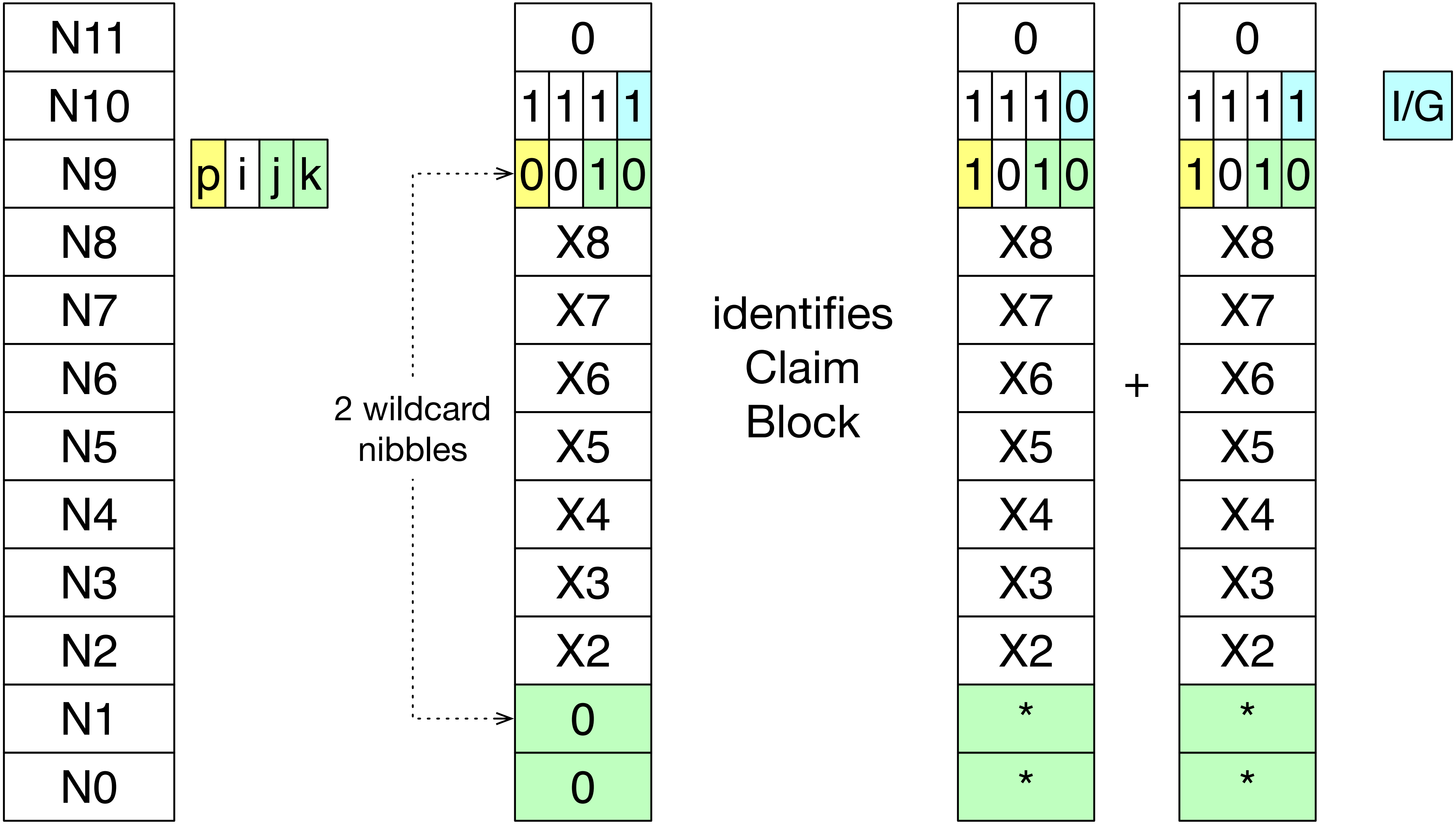


		identifier		1 unicast address		1 multicast address	
N11		0		0		0	
N10		1 1 1 1		1 1 1 0		1 1 1 1	I/G
N9	p i j k	0 0 0 0		1 0 0 0		1 0 0 0	
N8		X8		X8		X8	
N7		X7	identifies Claim Block	X7		X7	
N6		X6		X6	+	X6	
N5		X5		X5		X5	
N4		X4		X4		X4	
N3		X3		X3		X3	
N2		X2		X2		X2	
N1		X1		X1		X1	
N0		X0		X0		X0	

Claim Blocks and Identifiers (*jk*=01)



Claim Blocks and Identifiers (*jk*=10)



N11		0	0	0	
N10		1 1 1 1	1 1 1 0	1 1 1 1	I/G
N9	p i j k	0 0 1 0	1 0 1 0	1 0 1 0	
N8		X8	X8	X8	
N7		X7	X7	X7	
N6		X6	X6	X6	
N5		X5	X5	X5	
N4		X4	X4	X4	
N3		X3	X3	X3	
N2		X2	X2	X2	
N1		0	*	*	
N0		0	*	*	

2 wildcard nibbles

identifies Claim Block

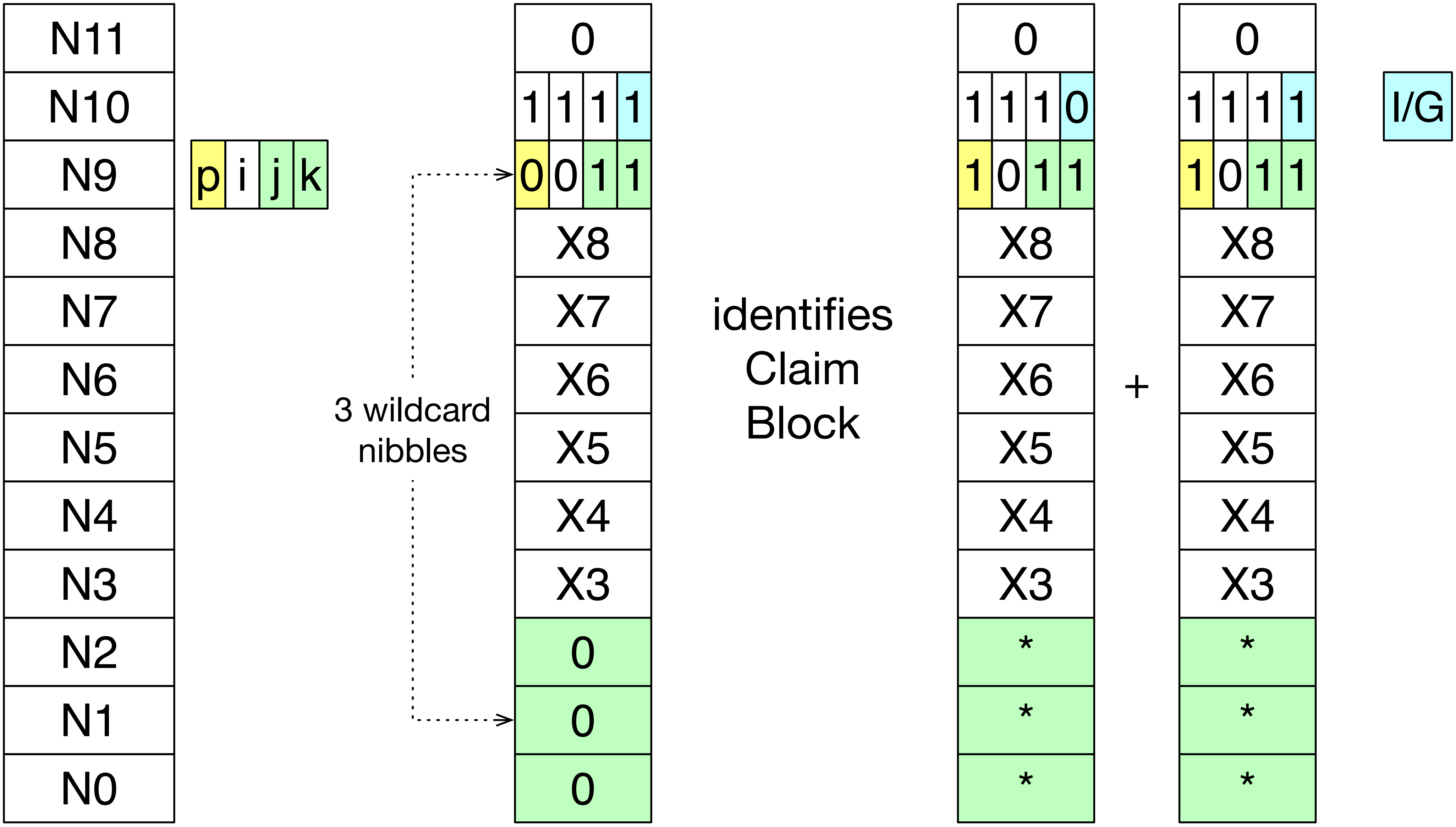
+

identifier

256 unicast addresses

256 multicast addresses

Claim Blocks and Identifiers (*jk*=11)



3 wildcard nibbles

identifies Claim Block

identifier

4096 unicast addresses

4096 multicast addresses

Claim Block Identifiers

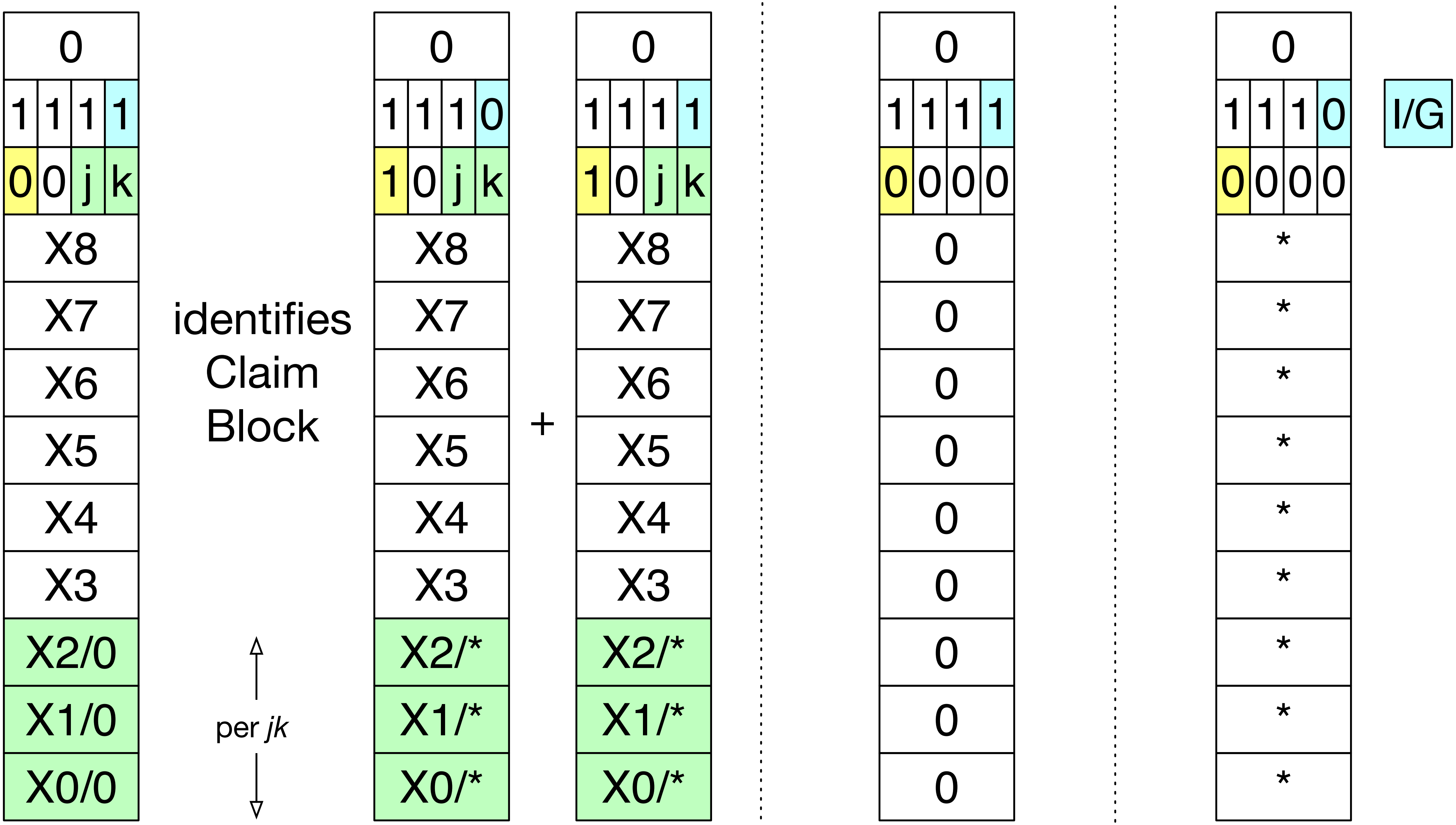
all blocks assignable simultaneously without address collision

	0	0	0	0	~~0~~	~~0~~	~~0~~	~~0~~
	1110	1110	1110	1110	~~1110~~	~~1110~~	~~1110~~	~~1110~~
	0	1	2	3	~~4~~	~~5~~	~~6~~	~~7~~
	X8	X8	X8	X8	~~X8~~	~~X8~~	~~X8~~	~~X8~~
	X7	X7	X7	X7	~~X7~~	~~X7~~	~~X7~~	~~X7~~
	X6	X6	X6	X6	~~X6~~	~~X6~~	~~X6~~	~~0~~
	X5	X5	X5	X5	~~X5~~	~~X5~~	~~0~~	~~0~~
	X4	X4	X4	X4	~~X4~~	~~0~~	~~0~~	~~0~~
	X3	X3	X3	X3	~~0~~	~~0~~	~~0~~	~~0~~
	X2	X2	X2	0	~~0~~	~~0~~	~~0~~	~~0~~
	X1	X1	0	0	~~0~~	~~0~~	~~0~~	~~0~~
	X0	0	0	0	~~0~~	~~0~~	~~0~~	~~0~~
block size*	1	16	256	4096	~~65536~~	~~1.0E6~~	~~1.6E7~~	~~2.7E8~~
block count*	6.9E10	4.3E9	2.7E8	1.6E7	~~1.0E6~~	~~65536~~	~~4096~~	~~256~~

*applies to both unicast and multicast blocks

larger block sizes probably not needed, so *i* bit of N9 can be reserved; could be used for free claiming (non-block claiming)

Claim Blocks and Identifiers



0			0		0	0	0
1 1 1 1			1 1 1 0		1 1 1 1	1 1 1 1	1 1 1 0
0 0 j k			1 0 j k		1 0 j k	0 0 0 0	0 0 0 0
X8			X8		X8	0	*
X7	identifies		X7		X7	0	*
X6	Claim		X6		X6	0	*
X5	Block		X5	+	X5	0	*
X4			X4		X4	0	*
X3			X3		X3	0	*
X2/0			X2/*		X2/*	0	*
X1/0	per *jk*		X1/*		X1/*	0	*
X0/0			X0/*		X0/*	0	*
identifier			contiguous unicast addresses		contiguous unicast addresses	null block identifier; identifies no block (heard by server)	non-claim temporary unicast source address

Properties of these Blocks and Identifiers

- Every block is identified by a unique block identifier.

- Every block identifier specifies a unique block.

- No address in a block is included in any other block.

- Address duplication between two different blocks, of any size, is impossible

- Each block includes unicast and multicast addresses
  - Same quantity of each.
  - All unicast addresses in a block are contiguous.
  - All multicast addresses in a block are contiguous.
  - Unicast and multicast address ranges in a block are identical except for the I/G bit.

Claim Block Examples

Example 1: device needs 1 address

- selects a claim, from the 6.9E+10 possible single blocks, by selecting 0–F for each of N0–N8.
- claim includes 1 unicast address and 1 matching multicast address

Example 2: device needs 12 addresses

- claims one wildcard nibble (N0)
- selects a claim, from the 2.6E+9 possible blocks, by selecting 0–F for each of N1–N8.
- claim includes 16 contiguous unicast addresses and 16 matching contiguous multicast addresses
  - –each with the selected N1–N8 values, and any value of N0

Example 3: device needs 200 addresses

- claims two wildcard nibbles (N0 and N1)
- selects a claim, from the 1.7E+8 possible blocks, by selecting 0–F for each of N2–N8.
- claim includes 256 contiguous unicast addresses and 256 matching contiguous multicast addresses
  - –each with the selected N2–N8 values, and any value of N0 and N1

Example 4: device needs 4000 addresses

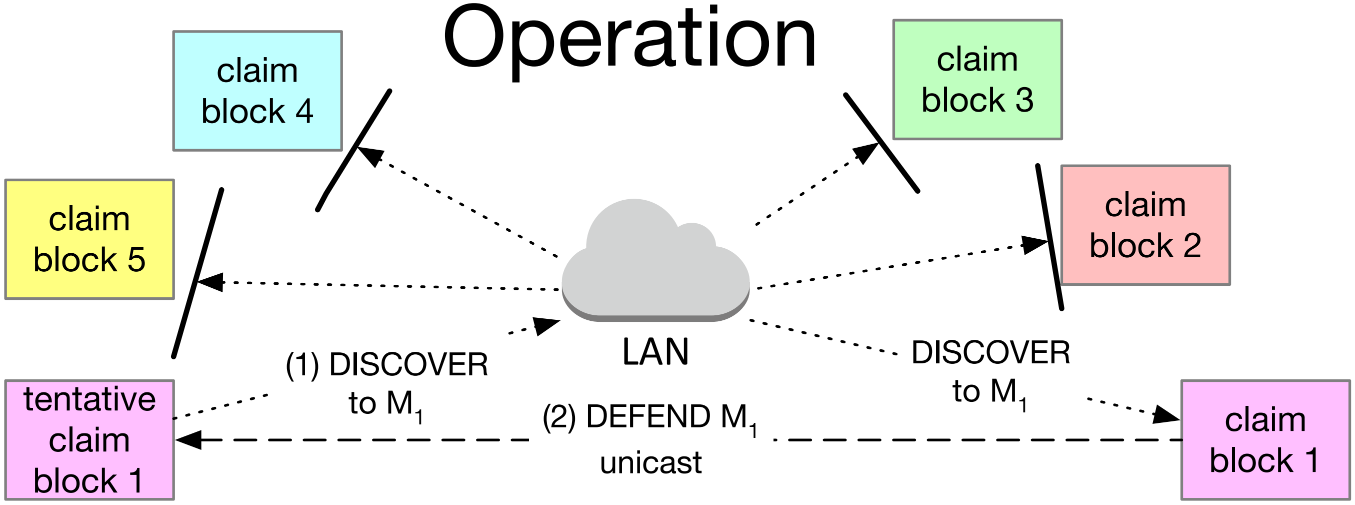
- claims three wildcard nibbles (N0, N1, N2)
- selects a claim, from the 1.1E+7 possible blocks, by selecting 0–F for each of N3–N8.
- claim includes 4096 contiguous unicast addresses and 4096 matching contiguous multicast addresses
  - –each with the selected N3–N8 values, and any value of N0, N1, N2

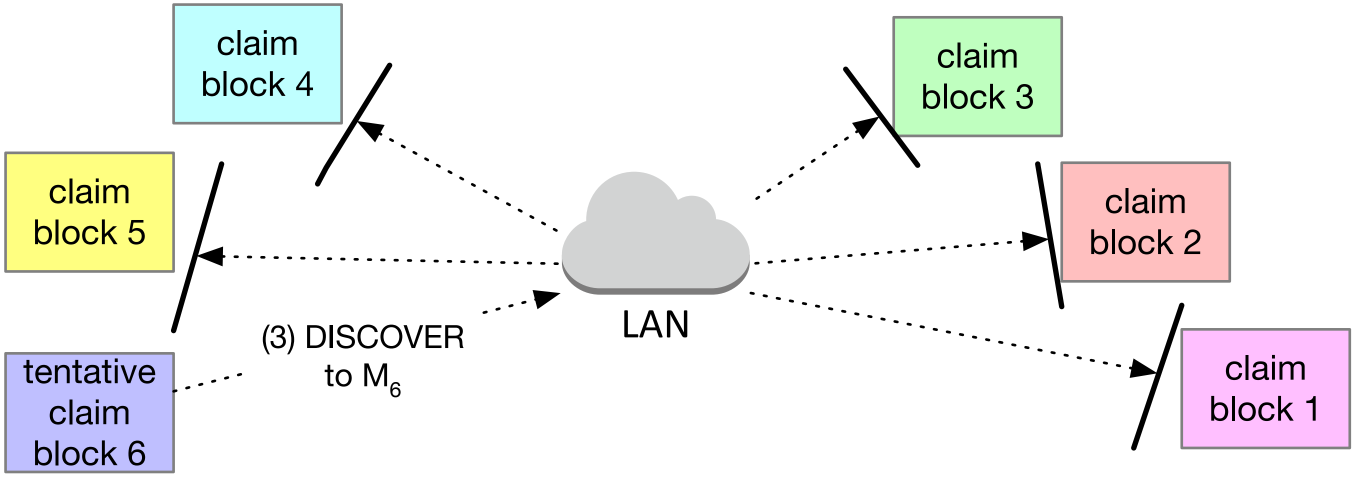
In each case,

–nibble value selections could be random, or some values could be reserved for specific purpose
e.g., an application

–claim could be limited to a subset (may not be worth the trouble)

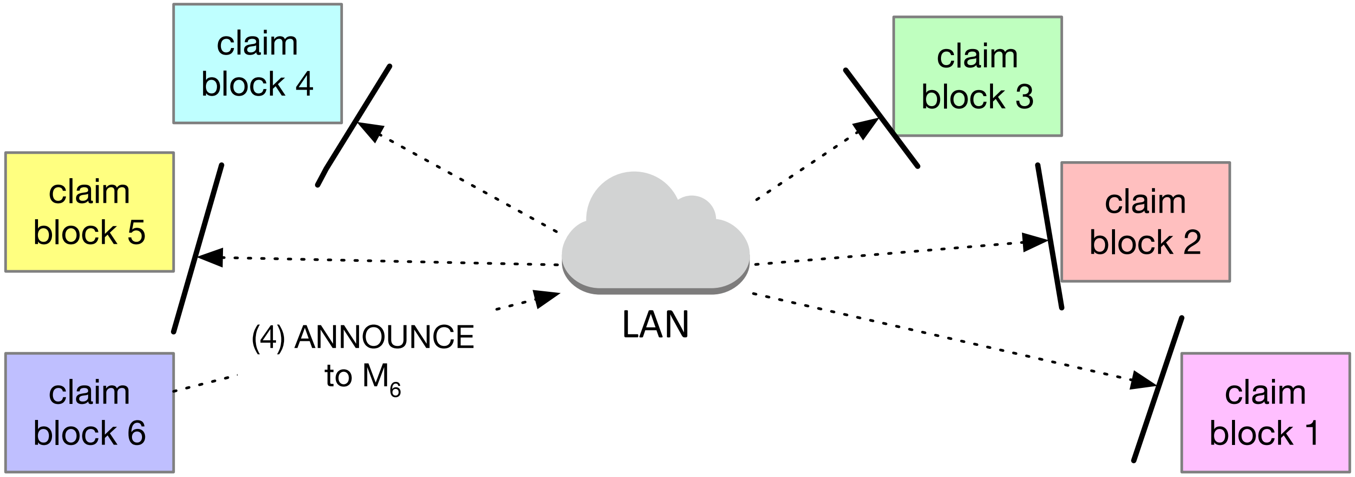
Operation



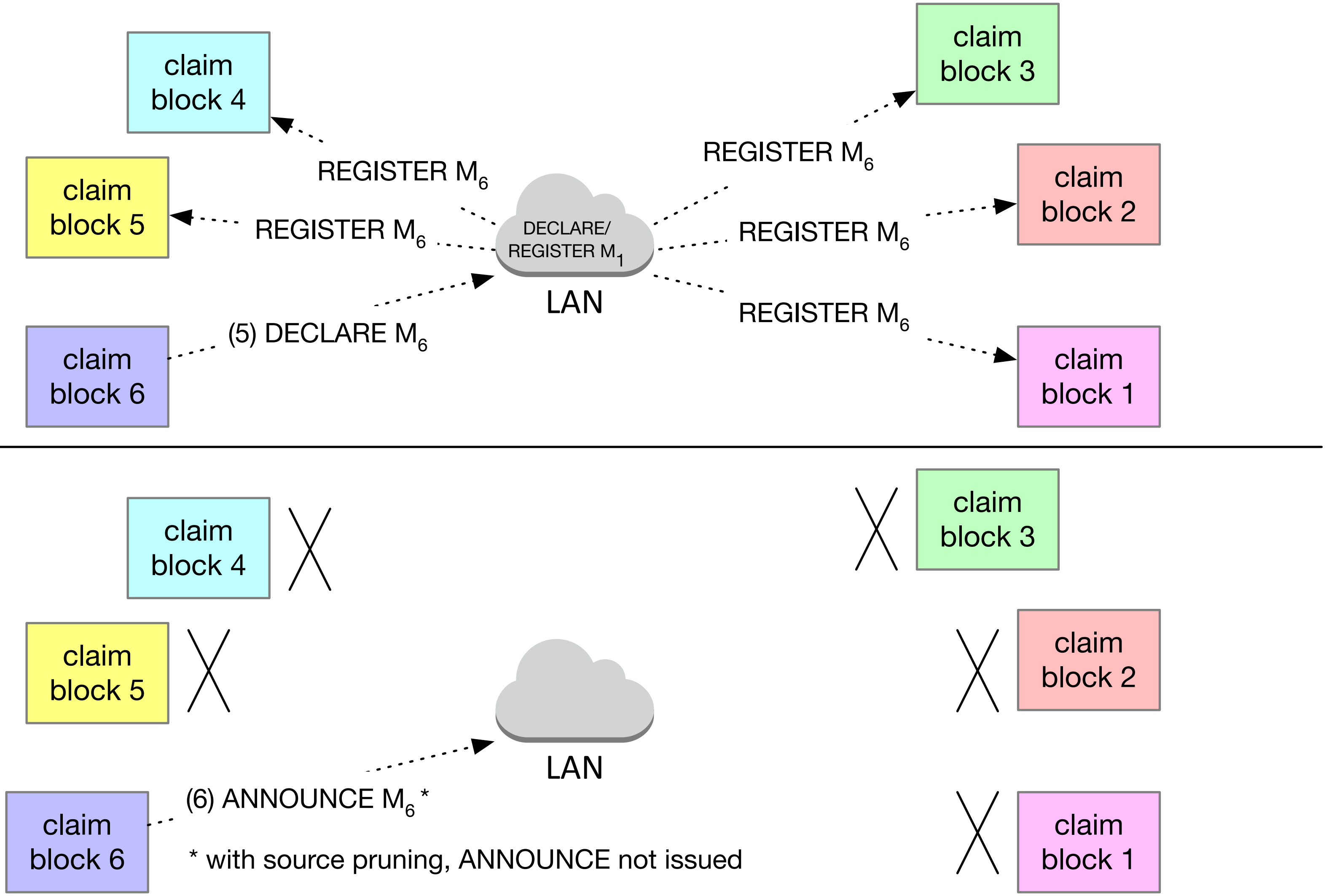


Note: Address server, if present, listens to all claiming messages (e.g. identifiable by initial bits 0000111100).

Alternatively, claiming messages could be repeated to null block identifier.



MMRP Declaration (after initial ANNOUNCE)



* with source pruning, ANNOUNCE not issued

Note: If a device includes an MMRP Participant, its filtering database should have a record of all registered claim block identifiers. By referring to that filtering database, the device should be able to choose an unused block, avoiding the duplicated claim in step (1) of the prior slide. This method would reduce collisions but is imperfect since filtering database may be obsolete at the time it is searched, particularly for a new device coming online (a likely case for an address claim) or in case of network merge. Perhaps registration data can be acquired from the network.

Server could declare to MMRP the range of all claim block identifiers.

Temporary Unicast Addresses

- Addresses beginning 0x0E0 are used for temporary, non-claim unicast addresses

- a device without a source address selects a random temporary address (starting 0x0E0) for DISCOVER/PROBE message <u>only</u>
    - – P802.1CQ then assigns at least one ongoing unicast address, using server assignment or claiming

- simultaneous duplicate temporary addresses may lead to loss of DEFEND or OFFER in some circumstances
    - – network learns route to source as DISCOVER/PROBE crosses the network
    - – before DEFEND or OFFER is returned, another DISCOVER crosses the path and rewrites the route
    - – unlikely to be disastrous
    - – loss of DEFEND or OFFER will be corrected eventually, upon ANNOUNCE or sooner

- problem ameliorated:
    - (1) DISCOVER/PROBE is repeated with a varying temporary address
    - (2) a duplicate after the DEFEND or OFFER is received is not a collision

- nevertheless, need to consider the likelihood of duplication

- Temporary address range includes 9 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 ~0.9999927
    - – address conflicts are rare, usually not harmful, and recoverable
    - – can add last three bits of the N9 nibble to the pool; chance of no duplicates is then ~0.99999909 (*k*=1000)

Address Usage

AVTPDU-range addresses (per IEEE Std 1722-2016)

N11	U/L	I/G	N10	from	to	PROBE/ ANNOUNCE DA
1001 (0x9)	0 (global)	1 (multicast)	0001 (0x1)	91-E0-F0-00-00-00	91-E0-F0-00-FD-FF	91-E0-F0-00-FF-00

local addresses

usage	N11	SLAP	U/L	I/G	N10	N9	DISCOVER/ ANNOUNCE DA
block claim	0000	SAI (11)	1 (local)	0 (unicast)	1110 (0xE)	10*jk*	0F-00*jk*-[block id]
block claim	0000	SAI (11)	1 (local)	1 (multicast)	1111 (0xF)	10*jk*	0F–00*jk*-[block id]
server	0001	SAI (11)	1 (local)	0 or 1	1110 or 1111	*	0F-00-00-00-00-00
server	*	AAI (00)	1 (local)	0 or 1	0010 or 0011	*	0F-00-00-00-00-00
server	per CID	ELI (01)	1 (local)	0 or 1	1010 or 1011	per CID	0F-00-00-00-00-00
consider:							
free claim	0000	SAI (11)	1 (local)	0 or 1	1110 or 1111	11**	0F-00-00-00-00-00

temporary unicast

usage	N11	SLAP	U/L	I/G	N10	N9
temp	0000	SAI (11)	1 (local)	0 (unicast)	1110 (0xE)	0000

Side Topic: MMRP Declaration of Unicast Block

MMRP discussion has considered declaration of claim identifiers.

What about MMRP declaration of claimed addresses?

For multicast, the device holding the assignment need not declare itself as a source. It might declare the listeners, if it knows them.

- Not directly related to P802.1CQ.

For unicast, it’s may be useful for the device holding the self-assignment to declare that assignment via MMRP, because:

(1) A one-step declaration covers a contiguous range of self-assigned addresses.

- Could even be integrated with MMRP declaration of multicast block identifier.

(2) Eliminates flooding for all the unicast addresses in the assignment.
(3) Eliminates the need for learning of each 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.

Summary

- Address duplication between any two of these claim blocks, of any size, is impossible.

- Benefits of discretized and identified claim blocks:
  - simple way to name claims and process claim conflicts
  - reduce interruptions to claiming devices
  - reduced multicast claiming traffic

- The proposed discretization of claim blocks is not the most efficient use of addresses, but
  - there are plenty of address available for a LAN
  - this discretization provides for operational efficiency and simplicity
  - each claim provides both unicast and multicast addresses
    - both have the same range, except for the I/G bit
      - might have some benefits
    - devices needing both addresses need make only one claim
    - claim objections are simplified

- Potential to integrate with MMRP to avoid the need for some announcements
  - MMRP needs to efficiently handle address ranges

- Null claim identifier address provided for server discovery and non-block claiming
  - a bit is available to specify a free-claim unicast and multicast pool within SAI range

- Range of addresses dedicated for use as temporary unicast addresses for all discovery and probe messaging.

Proposed Comment Resolution

- CID 46
    - Revise, “update draft per cq-marks-block-claims”