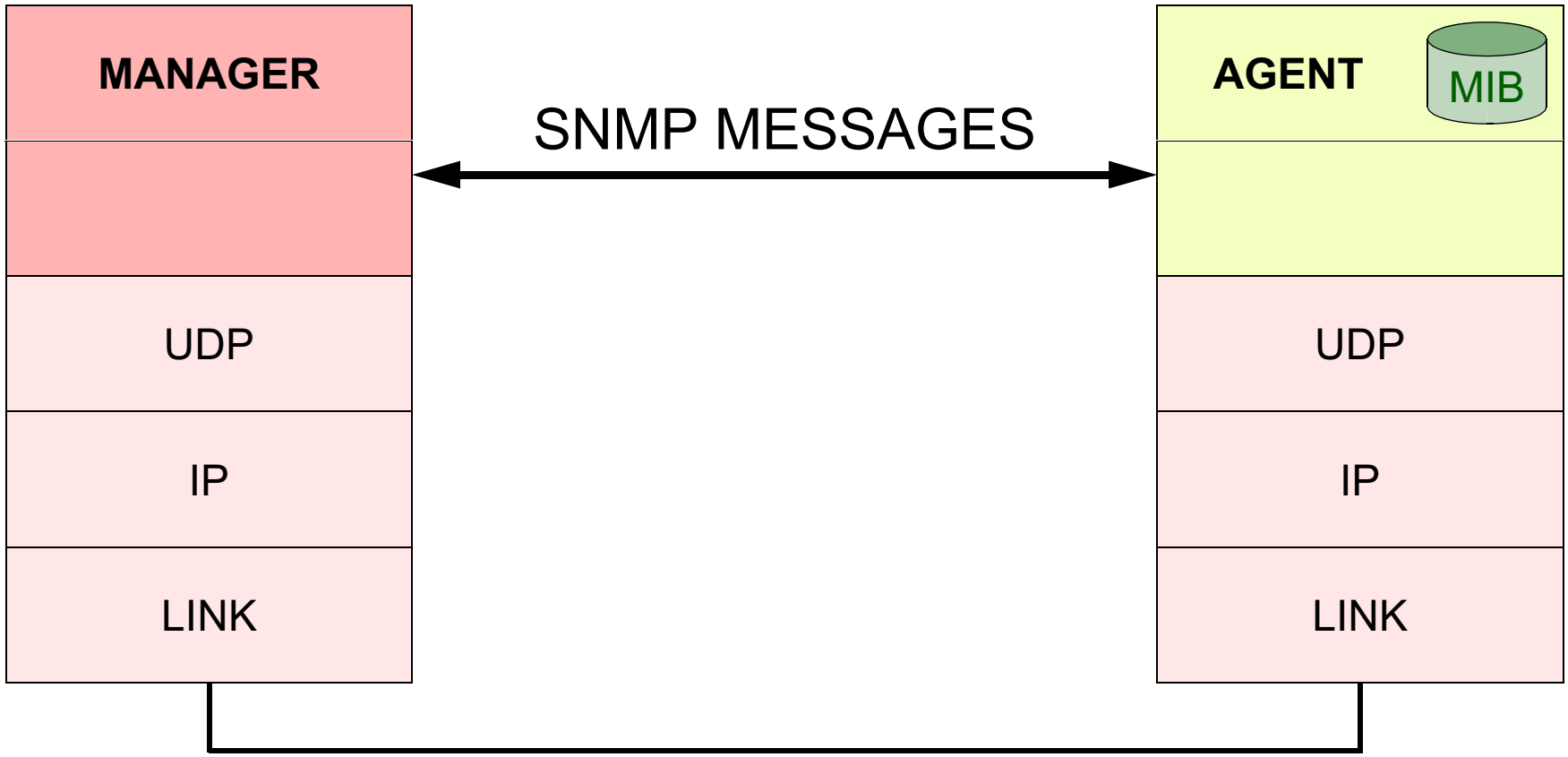
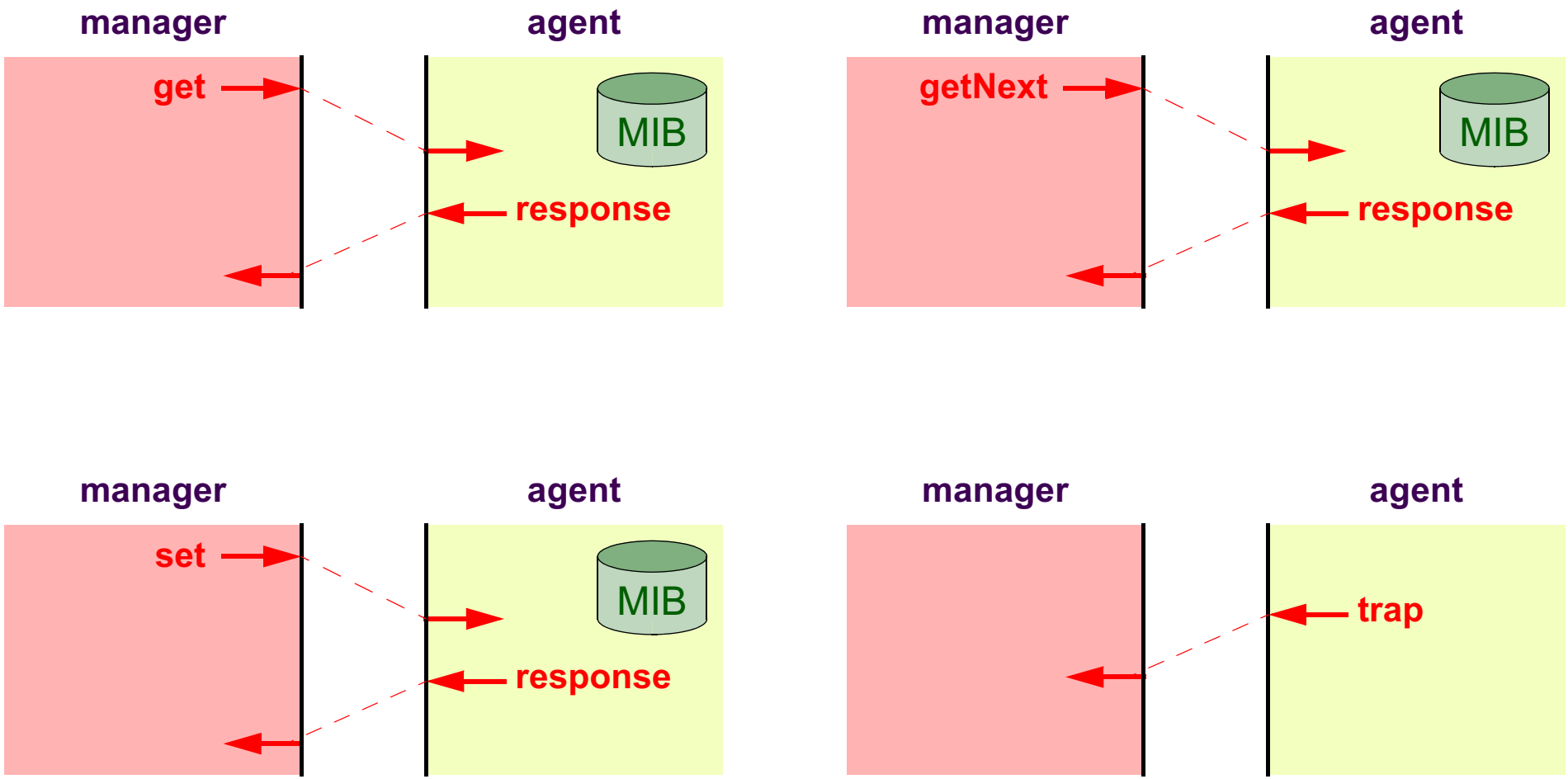
SNMPv1 PROTOCOL

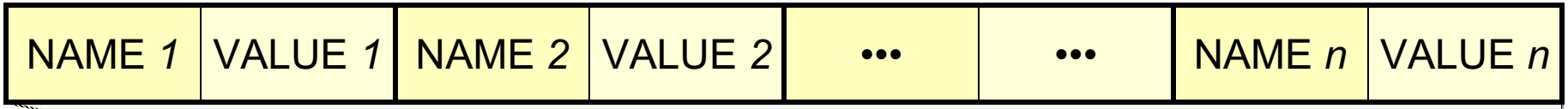


OVERVIEW OF PDUs



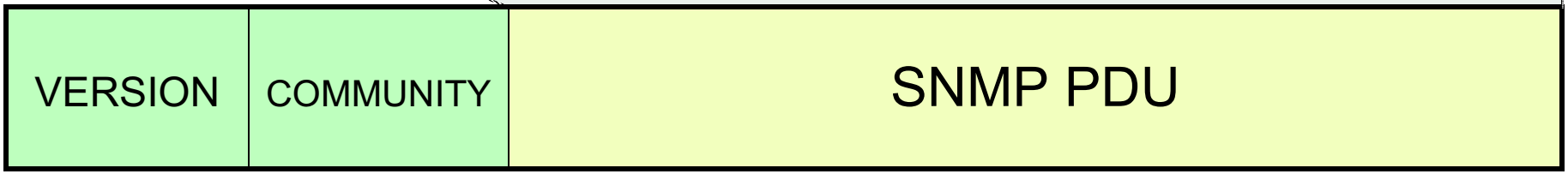
MESSAGE & PDU STRUCTURE

variable bindings:

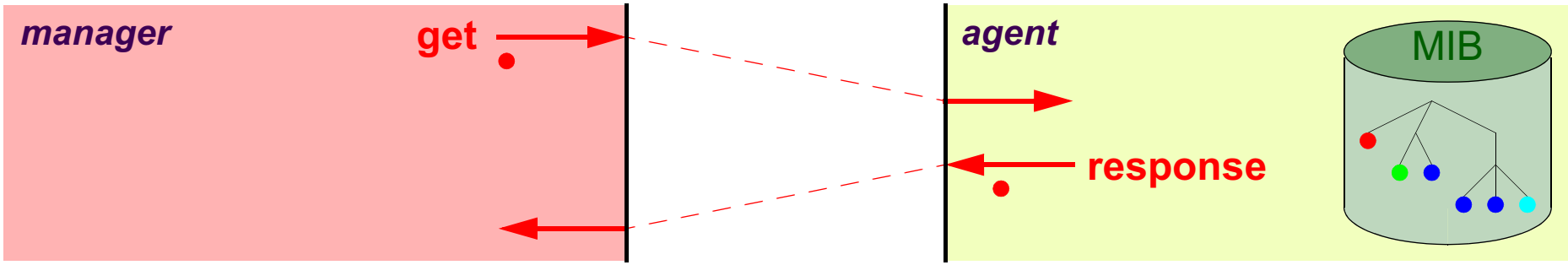


SNMP PDU:

SNMP message:



GET

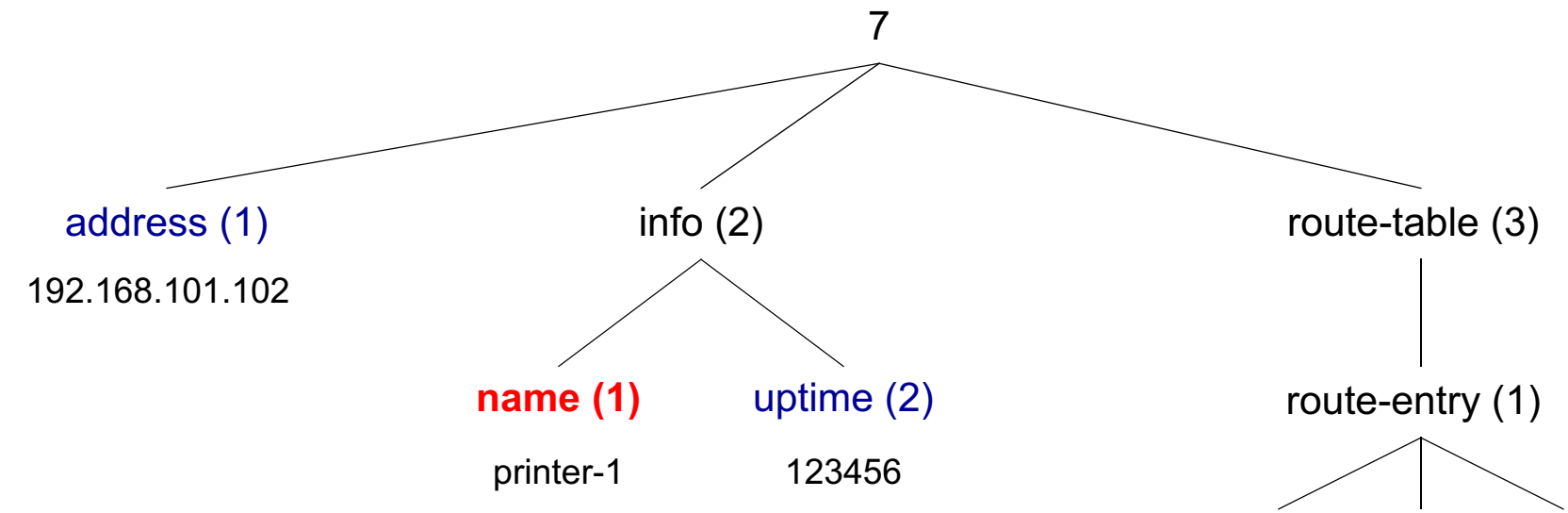


TO REQUEST THE VALUE OF 1 OR MORE VARIABLES

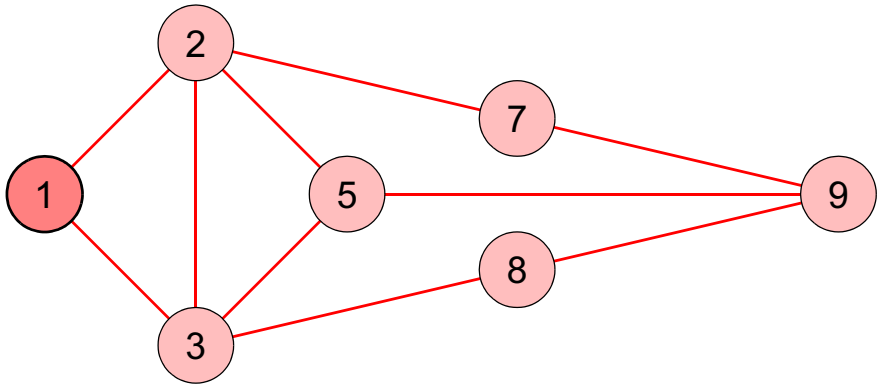
POSSIBLE ERRORS:

- noSuchName ⇨ Object does not exist / Object is not a leaf
- tooBig ⇨ Result does not fit in response PDU
- genErr ⇨ All other causes

EXAMPLE MIB



2	1	2
3	1	3
5	1	2
5	2	3
7	1	2
8	1	3
9	1	2



GET EXAMPLES

get(7.1.0)
response(7.1.0 => *192.168.101.102*)

get(7.2.0)
response(error-status = noSuchName)

get(7.1)
response(error-status = noSuchName)

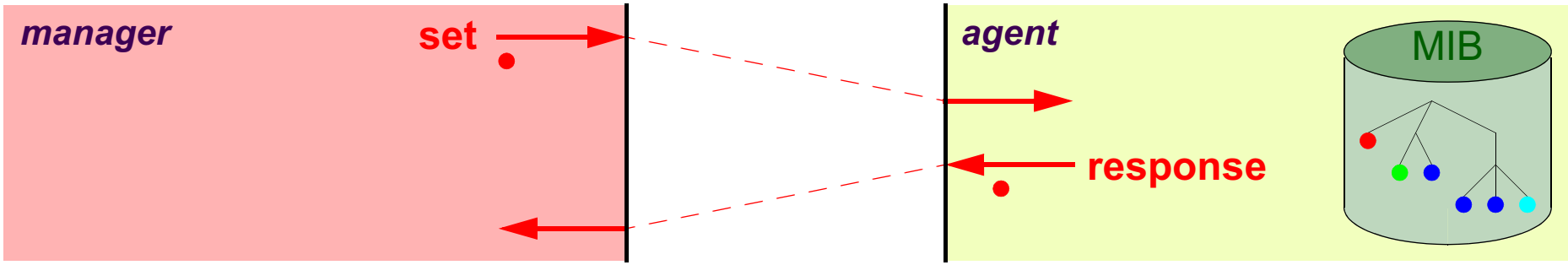
get(7.1.0; 7.2.2.0)
response(7.1.0 => *192.168.101.102*; 7.2.2.0 => *123456*)

get(7.3.1.3.5.1)
response(7.3.1.3.5.1 => *2*)

get(7.3.1.1.5.1)
response(7.3.1.1.5.1 => *5*)

get(7.3.1.1.5.1, 7.3.1.2.5.1, 7.3.1.3.5.1)
response(7.3.1.1.5.1 => *5*, 7.3.1.2.5.1 => *1*, 7.3.1.3.5.1 => *2*)

SET



TO ASSIGN A VALUE TO AN EXISTING OBJECT INSTANCE

TO CREATE NEW INSTANCES
- TABLE ROWS

THE SET REQUEST IS ATOMIC

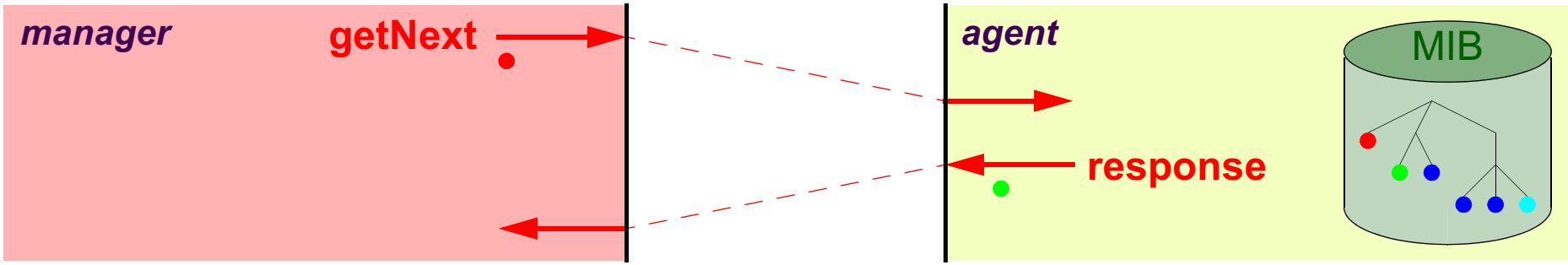
POSSIBLE ERRORS:
- noSuchName
- badValue
- tooBig
- genErr

SET EXAMPLES

set(7.2.1.0 => *my-printer*)
response(noError; 7.2.1.0 => *my-printer*)

set(7.2.1.0 => *my-printer*, 7.2.2.0 => *0*)
response(error-status = noSuchName; error-index = 2)

GET-NEXT



RETRIEVES THE INSTANCE NAME AND VALUE OF THE **NEXT** MIB ELEMENT

TO DISCOVER MIB STRUCTURES

TO RETRIEVE TABLE ROWS

POSSIBLE ERRORS:

- noSuchName (= END OF MIB)
- tooBig
- genErr

GET-NEXT EXAMPLES

getNext(7.1.0)
response(7.2.1.0 => *printer-1*)

getNext(7.2.1.0)
response(7.2.2.0 => *123456*)

getNext(1)
response(7.1.0 => *192.168.101.102*)

getNext(7.3.1.3.5.1)
response(7.3.1.3.5.2 => *3*)

getNext(7.3.1.1; 7.3.1.2; 7.3.1.3)
response(7.3.1.1.2.1 => *2*; 7.3.1.2.2.1 => *1*; 7.3.1.3.2.1 => *2*)

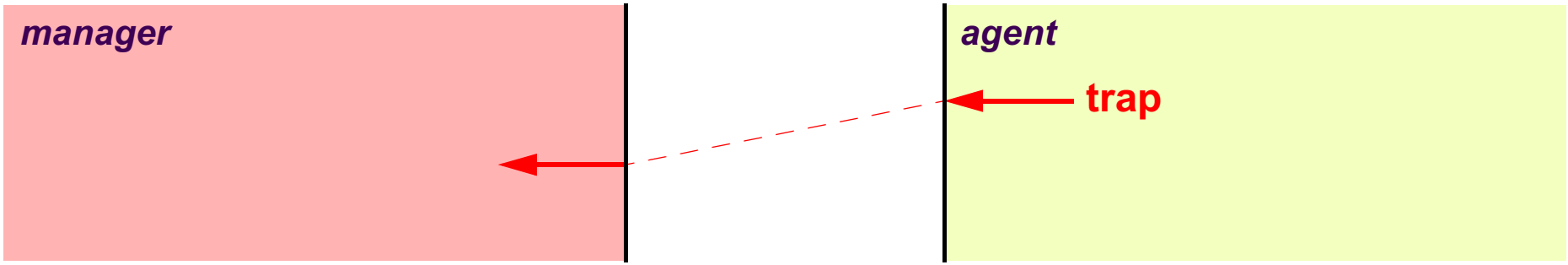
getNext(7.3.1.1.2.1; 7.3.1.2.2.1; 7.3.1.3.2.1)
response(7.3.1.1.3.1 => *3*; 7.3.1.2.3.1 => *1*; 7.3.1.3.3.1 => *3*)

LEXICOGRAPHICAL ORDERING

THE MIB CAN BE CONSIDERED AS AN ORDERED LIST

INSTANCE ID	INSTANCE VALUE
7.1.0	192.168.101.102
7.2.1.0	printer-1
7.2.2.0	123456
7.3.1.1.2.1	2
7.3.1.1.3.1	3
7.3.1.1.5.1	5
•••	•••
7.3.1.1.9.1	9
7.3.1.2.2.1	1
7.3.1.2.3.1	1
•••	•••
7.3.1.2.9.1	1
7.3.1.3.2.1	2
7.3.1.3.3.1	3
7.3.1.3.5.1	2
7.3.1.3.5.2	3
7.3.1.3.7.1	2
•••	•••

TRAP



TO SIGNAL AN EVENT

TRAP RECEPTION IS NOT CONFIRMED
(THUS UNRELIABLE)

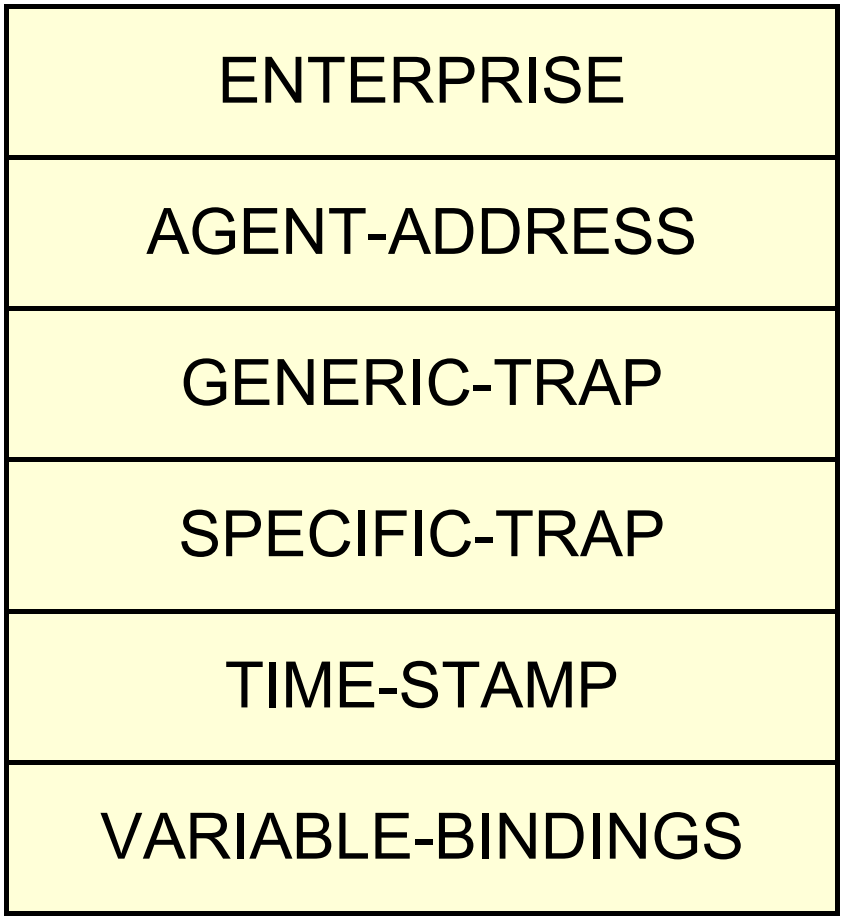
POLLING REMAINS NECESSARY

AGENTS MAY BE CONFIGURED SUCH THAT:
- NO TRAPS WILL BE TRANSMITTED
- TRAPS WILL BE TRANSMITTED TO CERTAIN MANAGERS

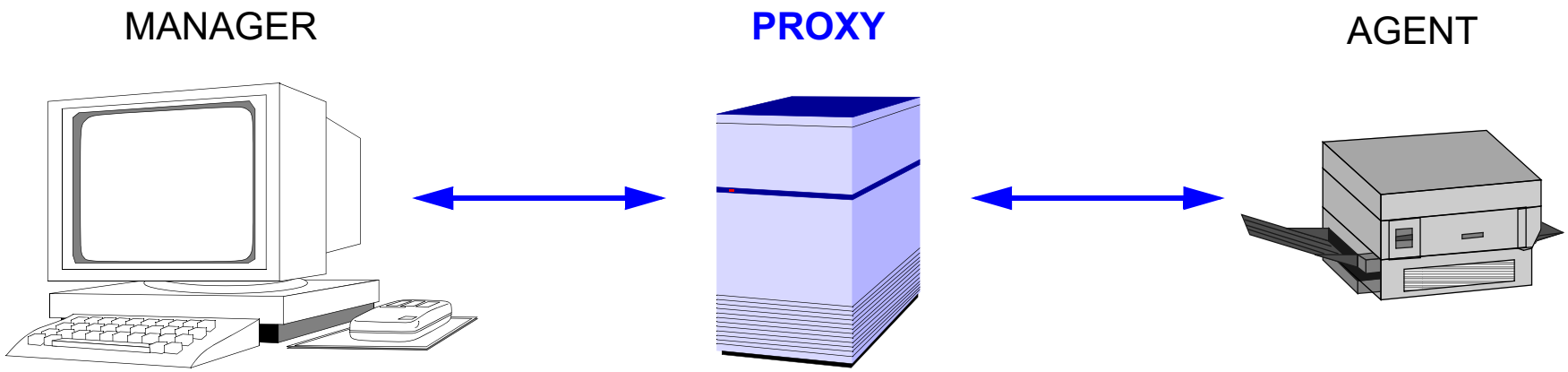
DEFINED TRAPS

- COLDSTART
- WARMSTART
- LINKDOWN
- LINKUP
- AUTHENTICATION FAILURE
- EGPNEIGHBOURLOSS
- ENTERPRISESPECIFICTRAP

TRAP - PDU FORMAT



PROXY MANAGEMENT

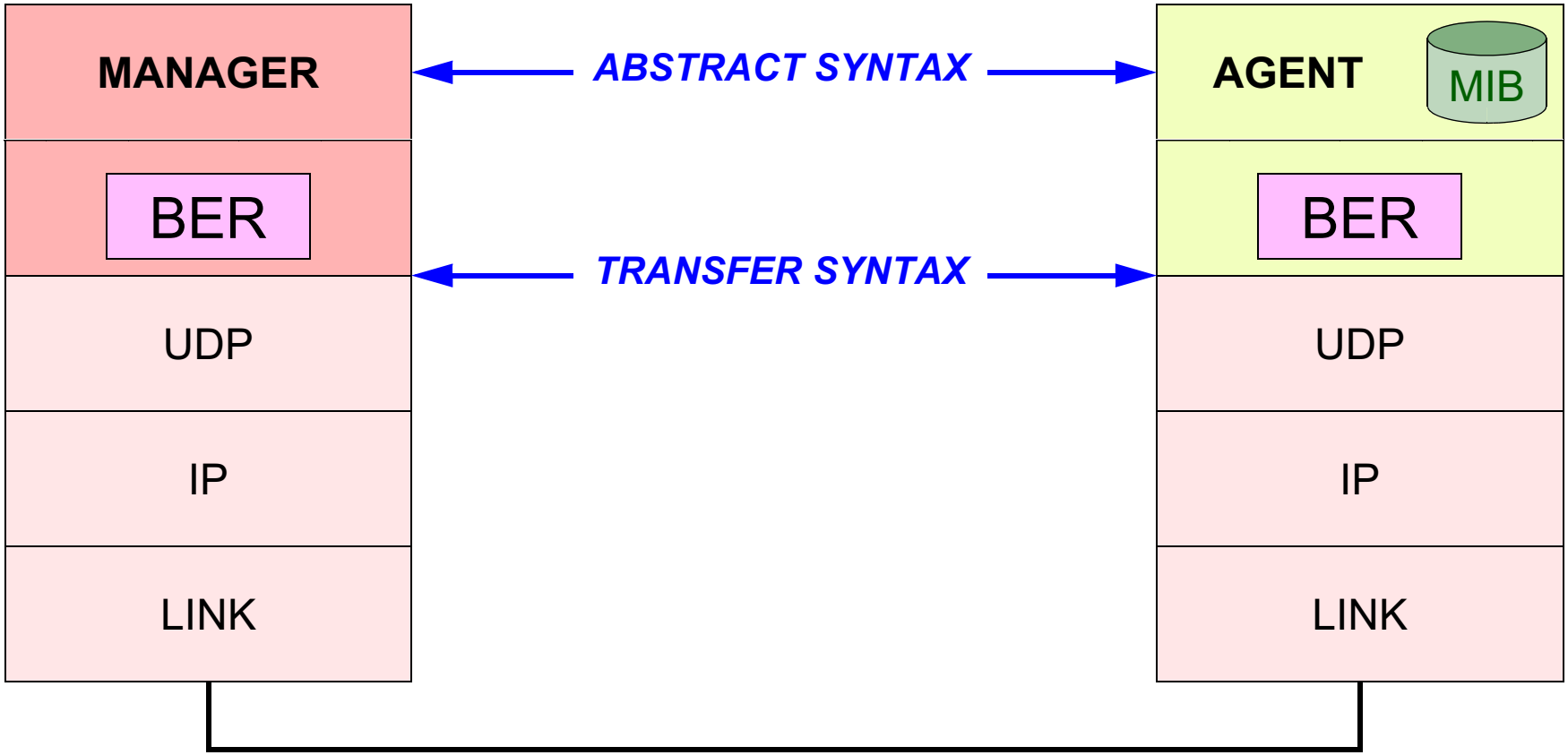


TERM HAS TRADITIONALLY BEEN USED FOR DEVICES THAT :

- TRANSLATE BETWEEN DIFFERENT TRANSPORT DOMAINS
- TRANSLATE BETWEEN DIFFERENT SNMP VERSIONS
- TRANSLATE BETWEEN SNMP AND OTHER MANAGEMENT PROTOCOLS
- AGGREGATE LOW LEVEL MANAGEMENT INFO INTO HIGH LEVEL INFO
- ETC.

NOWADAYS THE TERM DENOTES A DEVICE
THAT FORWARDS SNMP MESSAGES,
BUT DOESN'T LOOK AT THE INDIVIDUAL OBJECTS

SNMP MESSAGE ENCODING



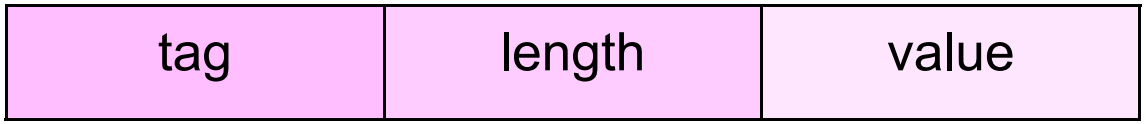
THE DESCRIPTION OF MIBS AND MESSAGE FORMATS
IS BASED ON THE ASN.1 SYNTAX

THE MAPPING FROM AN ABSTRACT SYNTAX UPON A TRANSFER SYNTAX
IS DEFINED BY THE BASIC ENCODING RULES (BER)

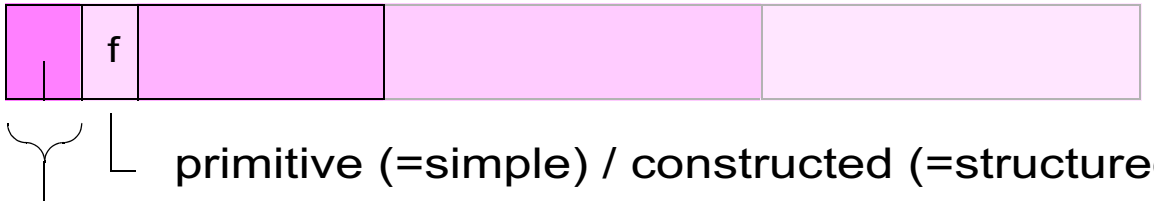
BASIC ENCODING RULES

EACH ASN.1 VALUE IS ENCODED AS AN OCTET STRING

THIS ENCODING RESULTS INTO A SEQUENCE OF
TAG, LENGTH, VALUE
STRUCTURES



TAG FIELD



Universal tags

BIT PATTERN	ASN.1 TYPE
00 0 0 0010	INTEGER
00 0 0 0100	OCTET STRING
00 0 0 0110	OBJECT IDENTIFIER

Application-wide tags

BIT PATTERN	APPLICATION TYPE
01 0 0 0000	IpAddress
01 0 0 0001	Counter32
01 0 0 0010	Gauge32
01 0 0 0010	Unsigned32
01 0 0 0011	TimeTicks
01 0 0 0100	Opaque
01 0 0 0110	Counter64

LENGTH FIELD

SHORT FORM:

LONG FORM: