Network Management

SNMP v2 & v3

12 May 2015

Overview – Lecture 2

- SNMPv2
- MIB-2 Enhancements
- SNMPv3 Framwork

RFC References – SNMPv2

- RFC 3416, “Version 2 of the Protocol Operations of the Simple Network Management Protocol”.
- RFC 2578, “Structure of Management Information Version 2 (SMIv2)”.
- RFC 2579, “Textual Conventions for SMIv2”.

SNMPv2 - Operations

Includes following SNMPv1 operations

- **Get**, retrieve specific objects
- **Get-Next**, retrieve objects by traversing a MIB tree
- **Set**, modify or create objects

In addition, the following operations are specified:

- **Get-Bulk**, for bulk retrieval of objects
- **SNMPv2 Trap**, traps can be specified using object identifiers
- **Inform**, acknowledged trap

SNMPv2 – Get Bulk

- Bulk retrieval of a table
- The Get Bulk request has a variable binding list and 2 additional values:
  - a non-repeaters value, and
  - a max-repetitions value
- A “GetBulkRequest (non-repeaters=1, max-repetitions=2, varbindlist={sysUpTime, ifInOctets})” will return:
  - sysUpTime.0 1233413
  - ifInOctets.1 345322
  - ifInOctets.2 9981
- If there is no lexicograhic successor, a value of “endOfMibView” will be returned

SNMPv2 Message – Get Bulk

SNMPv2 Get-Bulk PDU:

type	reqid	n	m	Variable bindings

type:
0xA5 – SNMPv2 Get-Bulk request

- n is the non-repeaters value.
- m is the max-repetitions value
- the first ‘n’ variables (in variable bindings) will be retrieved by a single get-next iteration.
- the rest of the variables will go through as many as m get-next iterations.

SNMPv2 – Trap

- SNMPv2 Trap PDU is identical to that of a Get, Get-Next, or Set PDU
- The variable bindings specify information about the trap
  - First variable provides agent's sysUpTime when the trap was generated
  - Second variable identifies what type of trap it is
  - Additional variables are based on the trap type
  - A variable bindings in a 'linkdown' trap will be:

sysUpTime.0	32216671
sysTrapOID.0	snmpTraps.3
ifIndex.4	4
ifAdminStatus.4	up
ifOperStatus.4	down

SNMPv2 Message - Trap

SNMPv2 Trap PDU:

type	reqid	0	0	Variable bindings

type:
0xA7 – SNMPv2 Trap

- Identical to that of a Get, Get-Next, and Set PDU
- Generic traps defined in SNMPv1 has object identifiers in SNMPv2

SNMPv2 Message - Inform

SNMPv2 Inform PDU:

type	reqid	0	0	Variable bindings

type:
0xA6 – SNMPv2 Inform request

- Identical to that of a Trap PDU
- Inform response PDU has the same request identifier and variable bindings

SMIv2

- The SMI is divided into three parts:
  - Module definitions are used to describe MIB modules using MODULE-IDENTITY macro.
  - Object definitions are used to describe managed objects. An OBJECT-TYPE macro is used to convey the syntax and semantics of a managed object.
  - Notification definitions are used to describe unsolicited transmissions of management events. A NOTIFICATION-TYPE macro is used to convey the syntax and semantics of a notification.
- Defined in RFC 2578

SMIv2 Module Identity

```
ifMIB MODULE-IDENTITY
LAST-UPDATED "200006140000Z"
ORGANIZATION "IETF Interfaces MIB Working Group"
CONTACT-INFO " Keith McCloghrie Cisco Systems, Inc. 170
    West Tasman Drive San Jose, CA 95134-1706 US
    408-526-5260 kzm@cisco.com"
DESCRIPTION "The MIB module to describe generic objects
    for network interface sub-layers. This MIB is an
    updated version of MIB-II's ifTable, and incorporates
    the extensions defined in RFC 1229."
REVISION "200006140000Z"
DESCRIPTION "Clarifications agreed upon by the Interfaces
    MIB WG, and published as RFC 2863."
REVISION "199602282155Z"
DESCRIPTION "Revisions made by the Interfaces MIB WG, and
    published in RFC 2233."
::= { mib-2 31 }
```

SMIv2 Managed Object Definition

- A textual name along with its OBJECT IDENTIFIER.
- SYNTAX clause
  - Adds BITS construct: represents enumeration of named bits.
- DESCRIPTION clause
- MAX-ACCESS replacing ACCESS clause
  - One of read-only, read-write, read-create, not-accessible or accessible-for-notify.
- STATUS clause
  - One of current, obsolete or deprecated
- NOTIFICATION-TYPE Macro: used for defining SNMPv2 Traps.

SMIv2 Managed Object Definition (contd.)

- AUGMENTS clause
  - used to add additional columns to an existing table.
- UNITS clause (optional)
  - contains the textual definition of the units associated with an object.
- REFERENCE (optional)
  - contains textual cross reference to an external MIB object.
- DEFVAL (optional)
  - provides a default value for the MIB object

SMIv2 – Augments Clause

```
ifXTable OBJECT-TYPE
SYNTAX SEQUENCE OF IfXEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION "A list of interface entries. The number of
   entries is given by the value of ifNumber. This table
   contains additional objects for the interface table."
::= { ifMIBObjects 1 }

ifXEntry OBJECT-TYPE
SYNTAX IfXEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION "An entry containing additional management
   information applicable to a particular interface."
AUGMENTS { ifEntry }
::= { ifXTable 1 }
```

SMIv2 – Augments Clause (contd.)

```
IfXEntry ::= SEQUENCE {
        ifName DisplayString,
        ifInMulticastPkts Counter32,
        ifInBroadcastPkts Counter32,
        ifOutMulticastPkts Counter32,
        ifOutBroadcastPkts Counter32,
        ifHCInOctets Counter64,
        ifHCInUcastPkts Counter64,
        ifHCInMulticastPkts Counter64,
        ifHCInBroadcastPkts Counter64,
        ifHCOutOctets Counter64,
        ifHCOutUcastPkts Counter64,
        ifHCOutMulticastPkts Counter64,
        ifHCOutBroadcastPkts Counter64,
        ifLinkUpDownTrapEnable INTEGER,
        ifHighSpeed Gauge32,
        ifPromiscuousMode TruthValue,
        ifConnectorPresent TruthValue,
        ifAlias DisplayString,
        ifCounterDiscontinuityTime TimeStamp }
```

SMIv2 – BITS Construct

- Enumeration of named bits
  - SYNTAX BITS { up (0), down (1), testing (2),reserved (3)}
- Assigned non-negative, contiguous value, starting at zero
- Only enumerated bits are present
- Recommended no more than 128 bits used
- Labels for the bits – up to 64 characters

SMIv2 – BITS Construct

```
pmSchedWeekDay OBJECT-TYPE
SYNTAX BITS { sunday(0), monday(1), tuesday(2),
  wednesday(3), thursday(4), friday(5), saturday(6) }
MAX-ACCESS read-create
STATUS current
DESCRIPTION
  "Within the overall time period specified in the
  pmSchedTimePeriod object, the value of this object
  specifies the specific days of the week within that
  time period when the schedule is active. Setting
  all bits will cause the schedule to act
  independently of the day of the week."
DEFVAL { { sunday, monday, tuesday, wednesday,
  thursday, friday, saturday } }
::= { pmSchedEntry 7 }
```

SMIv2 – Notification Type

```
linkDown NOTIFICATION-TYPE
OBJECTS {ifIndex, ifAdminStatus, ifOperStatus}
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A linkDown trap signifies that the SNMPv2 entity,
acting in an agent role, has detected that the
ifOperStatus object for one of its communication
links is about to transition into the down state
from some other state. This other state is
indicated by the included value of ifOperStatus."

::= { snmpTraps 3 }
```

SMIv2 Abstract Data Types

- SMIv2 defines the following additional data types:
  - Integer32
    - Same as an INTEGER
  - Counter32
    - Same as a Counter
  - Gauge32
    - Same as a Gauge
  - Unsigned32
    - Same as a Gauge32
  - Counter64
    - IMPLICIT INTEGER
    - 64-bit non-negative integer (0..2^{64}-1)
    - Used to measure values that wrap in less than an hour with 32-bit counters.

SMIv2 – Textual Conventions

- DisplayString
  - Resolves into an OCTET STRING
  - Management information that represents text is typically defined as a DisplayString
- PhysAddress
  - Resolves into an OCTET STRING
  - Represents a media or physical level address
- MacAddress
  - Resolves into a 6-byte OCTET STRING
  - Represents an 802 MAC address
- TruthValue
  - Resolves into an INTEGER
  - Represents to a boolean value of either true (1) or false (2)

SMIv2 – Textual Conventions

- TestAndIncr
  - Resolves into an INTEGER
  - Used to prevent 2 or more management stations from trying to modify the same management information at the same time.
- AutonomousType
  - Resolves to an OBJECT IDENTIFIER
  - Define a MIB sub-tree or a particular type of hardware or protocol
- VariablePointer
  - Resolves into an OBJECT IDENTIFIER
  - Pointer to a specific object instance
- RowPointer
  - Resolves into an OBJECT IDENTIFIER
  - Pointer to a row

SMIv2 – Textual Conventions

- TimeStamp
  - Resolves to TimeTicks
  - Value of sysUpTime when a specific occurrence happened
- TimeInterval
  - Resolves to an INTEGER
  - Represents a period of time measured in hundredths of seconds
  - Maximum value of 248 days (21474836.47 seconds)
- DateAndTime
  - Resolves into an OCTET STRING (8- or 11-byte)
  - Represents a date-time specification
  - Display format: “2d-1d-1d,1d:1d:1d.1d,1a1d:1d”
  - Jan 15, 1999, 1:30:15 PM EDT would be displayed as
    - 1999-1-15,13:30:15.0,-4:0

SMIv2 – Textual Conventions

- TAddress
  - Resolves to an OCTET STRING
  - Represents a transport address
    - UDP Domain – 6-byte OCTET STRING (IP Address, UDP Port)
- StorageType
  - Resolves to an INTEGER
  - Standard way to specify how a row should be stored in memory
  - Enumerated:
    - other (1)
    - volatile (2)
    - nonVolatile (3)
    - permanent (4)
    - readOnly (5)

SMIv2 – Textual Conventions

- RowStatus
  - Resolves into an INTEGER
  - Provides a standard way to add or delete rows from a table
  - Enumerated
    - active (1)
    - notInService (2)
    - notReady (3)
    - createAndGo (4)
    - createAndWait (5)
    - destroy (6)

SMIv2 – RowStatus

The RowStatus textual convention is used to manage the creation and deletion of conceptual rows and has six defined values:

1) `active‘ - the row is available for use by the managed device
2) `notInService‘ - the row exists in the agent, but is unavailable for use by the managed device
3) `notReady‘ - the row exists in the agent, but is missing information necessary in order to be available for use by the managed device (i.e., one or more required columns in the row have not been instantiated)
4) ‘createAndGo‘ - written by a manager to create a new instance of a row and to have its status automatically set to active, making it available for use by the managed device
5) ‘createAndWait‘ - written by a manager to create a new instance of a row but not make it available for use by the managed device
6) ’destroy’ - written by a manager to delete the existing row.

SMIv2 – RowStatus

- When queried, an existing row has three states:
  – ‘active’ – row available for use by the managed device,
  – ‘notInService’ - row not available for use by the managed device, though the agent has sufficient information, or,
  – ‘notReady’ - row not available for use by the managed device because the agent has insufficient information.

Section 2
IETF MIB-2 Enhancements

MIB-II Groups' Newer Versions

- System & SNMP Groups
  - SNMPv2 MIB (RFC 3418)
- Interfaces Group
  - IF-MIB (RFC 2863),
- IP Group
  - IP-MIB (RFC 4293)
  - IP-FORWARD-MIB (RFC 2096)
- TCP Group
  - TCP-MIB (RFC 4022)
- UDP Group
  - UDP-MIB (RFC 4113)

System Group Changes

- Defined in RFC 3418
- Contains a new table object, sysORTable, and a new scalar object, sysORLastChange
  - sysORTable ::= {system 9}, describes SNMPv2 entity's support of various MIB modules and has 4 objects defined.
    - sysORIndex,
    - sysORID,
    - sysORDescr, and
    - sysORUpTime
  - sysORLastChange ::= {system 8}, contains the value of sysUpTime when sysORTable was last changed.

SNMP Group Changes

- Defined in RFC 3418
- Redefines original mib-2 SNMP group using SMIv2.
- Defines ‘snmpTrap’ group with 2 objects:
  - snmpTrapOID – object identifier of the trap being sent (second varbind in SNMPv2 Trap or Inform Request PDU).
  - snmpTrapEnterprise – object identifier of the enterprise associated with the trap being sent (last varbind in SNMPv2 Trap or Inform Request PDU).
- Defines ‘snmpTraps’ group to map SNMPv1 generic traps

SNMP Group Changes

- snmpTrap ::= internet.6.3.1.1.4
- snmpTrapOID ::= {snmpTrap 1}
- snmpTrapEnterprise ::= {snmpTrap 3}
- snmpTraps ::= internet.6.3.1.1.5
  - coldStart ::= {snmpTraps 1}
  - warmStart ::= {snmpTraps 2}
  - linkDown ::= {snmpTraps 3}
  - linkUp ::= {snmpTraps 4}
  - authenticationFailure ::= {snmpTraps 5}
  - egpNeighborLoss ::= {snmpTraps 6}

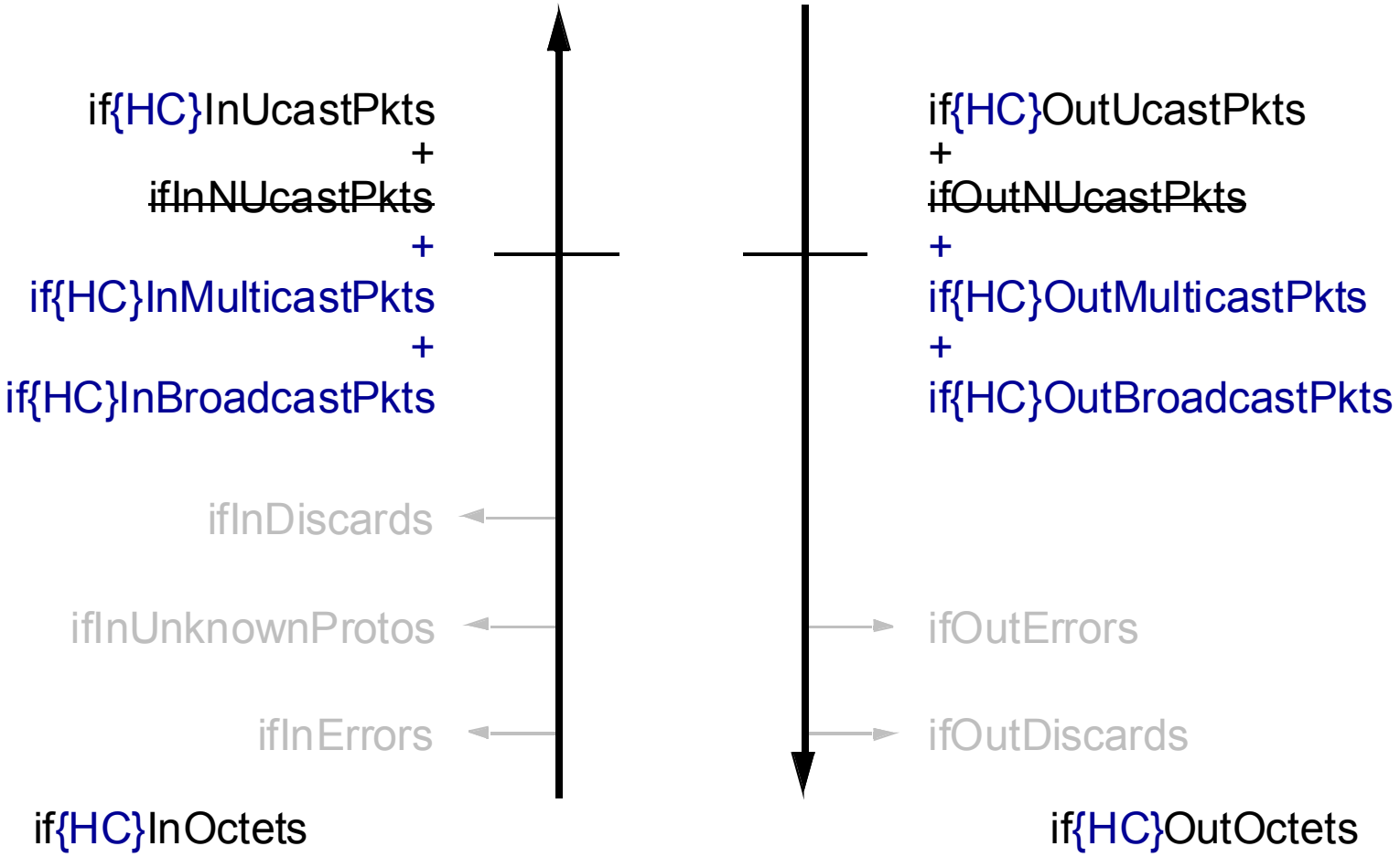
IF-MIB

- RFC 2863
- Rewrites IF group of MIB-II using SMIv2
- ifMib ::= {mib-2 31}
- ifMibObjects ::= {ifMib 1}
- Defines 4 tables:
  - ifTable ::= {interfaces 2}
  - ifXTable ::= {ifMibObjects 1}
  - ifStackTable ::= {ifMibObjects 2}
  - ifRcvAddressTable ::= {ifMibObjects 4}

IF-MIB

- ifTable
  - ifType has been redefined to be of IANAifType.
  - Interfaces not restricted to 1 through the value of ifNumber.
  - Adds a new scalar object ‘ifTableLastChange’
- ifXTable
  - Extension to the ifTable.
  - Additional management information pertaining to a particular interface.
  - Separate broadcast and multicast counters.
  - High Capacity 64-bit counters.
  - Supports interface speeds greater than 2.2 Gbps
  - Deprecates several objects in the ifTable (ifInNUcastPkts, ifOutNUcastPkts, ifSpecific, ifOutQLen)

ifXTable – 64-bit Counters



ifStackTable

- Allows virtual interfaces to be layered on top of physical interfaces.
- Shows the relationship between different sub-layers of network interfaces.
- It shows which sub-layers run on top of which other sub-layers, where each sub-layer corresponds to a conceptual row in the ifTable.
- ifStackHigherLayer – value of ifIndex corresponding to the higher sub-layer
- ifStackLowerLayer – value of ifIndex corresponding to the lower sub-layer

Section 3
SNMPv3

SNMPv3 - RFCs

- RFC 3411, “An Architecture for Describing Simple Network Management Protocol Management Frameworks”.
- RFC 3412, “Message Processing and Dispatching for the Simple Network Management Protocol”.
- RFC 3413, “Simple Network Management Protocol Applications”.

SNMPv3 Security Requirements

The **principal threats** against which a Security Model should protect include:

- **Modification of Information**

  The modification threat of in-transit SNMP messages by an unauthorized entity to effect unauthorized management operations.

- **Masquerade**

  Unauthorized management operations may be attempted by assuming the identity of another principal that has the appropriate authorizations.

SNMPv3 Security Requirements

The **secondary threats** against which a Security Model should provide protection include:

- **Message Stream Modification**

  Messages may be maliciously re-ordered, delayed or replayed to an extent to effect unauthorized management operations.

- **Disclosure**

  Eavesdropping on the exchanges between SNMP engines.

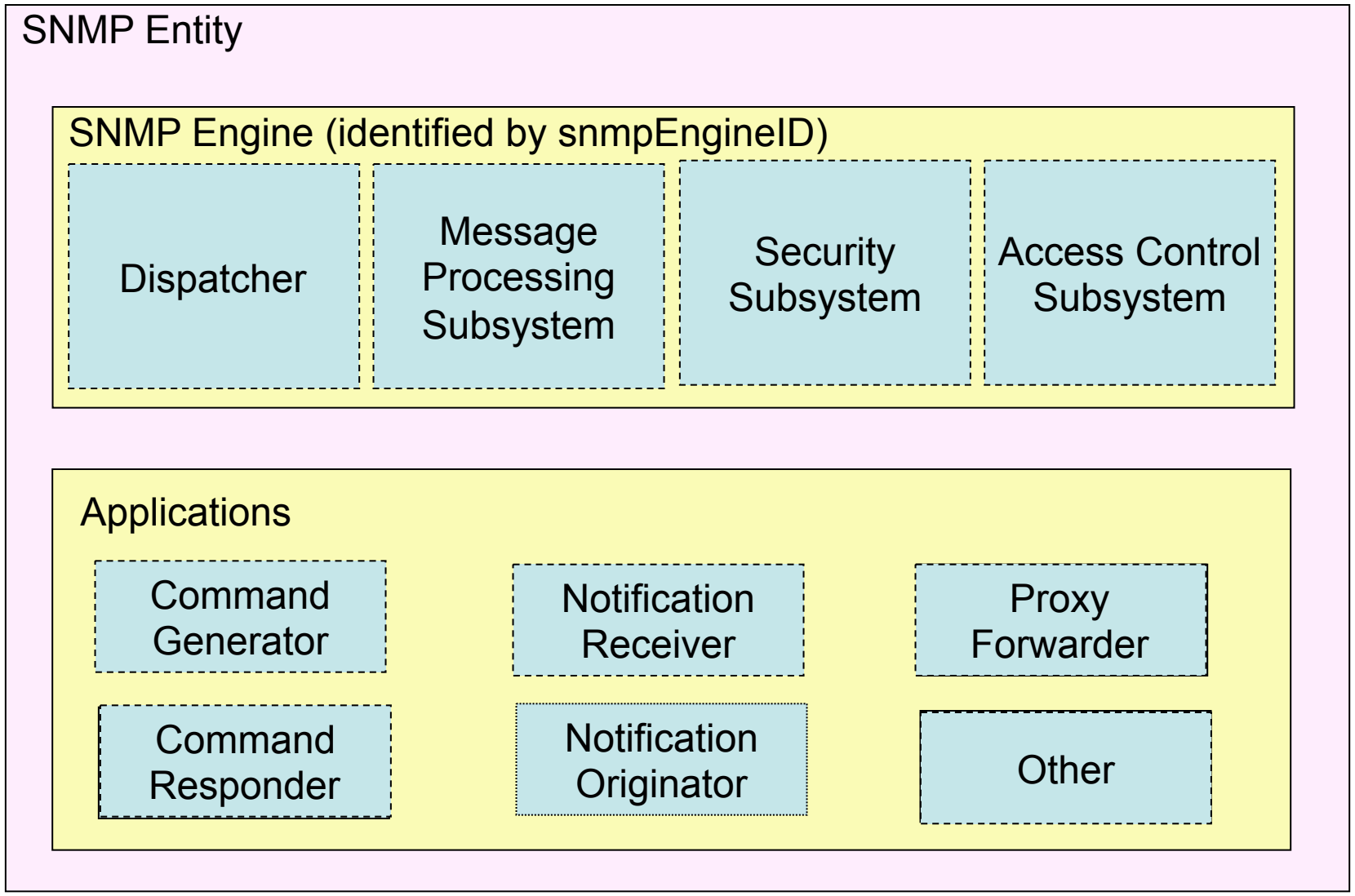
SNMPv3 Management Framework

- SNMPv3 is SNMPv2 plus Administration and Security.
- Proposes a modular architecture that can be easily extended.
- Consists of :
  - SNMP entities
    - a SNMP engine
    - a set of applications
  - a new message format
  - a security model, and
  - an access control model.

SNMP Entity

- Each SNMP entity consists of an SNMP engine and one or more associated applications
- There is a one-to-one association between an SNMP engine and the SNMP entity which contains it.

SNMP Entity


SNMP Entity
SNMP Engine (identified by snmpEngineID)
Dispatcher
Message Processing Subsystem
Security Subsystem
Access Control Subsystem
Applications
Command Generator
Notification Receiver
Proxy Forwarder
Command Responder
Notification Originator
Other

SNMP Engine

- An SNMP Engine provides services for:
  - sending and receiving messages
  - authenticating and encrypting/decrypting messages, and
  - controlling access to managed objects.
- SNMP Engine consists of the following subsystems:
  - Dispatcher Subsystem
  - Message Processing Subsystem
  - Security Subsystem
  - Access Control Subsystem
- ‘snmpEngineID’ is the unique and unambiguous identifier of both an SNMP Entity and the SNMP Engine it contains.

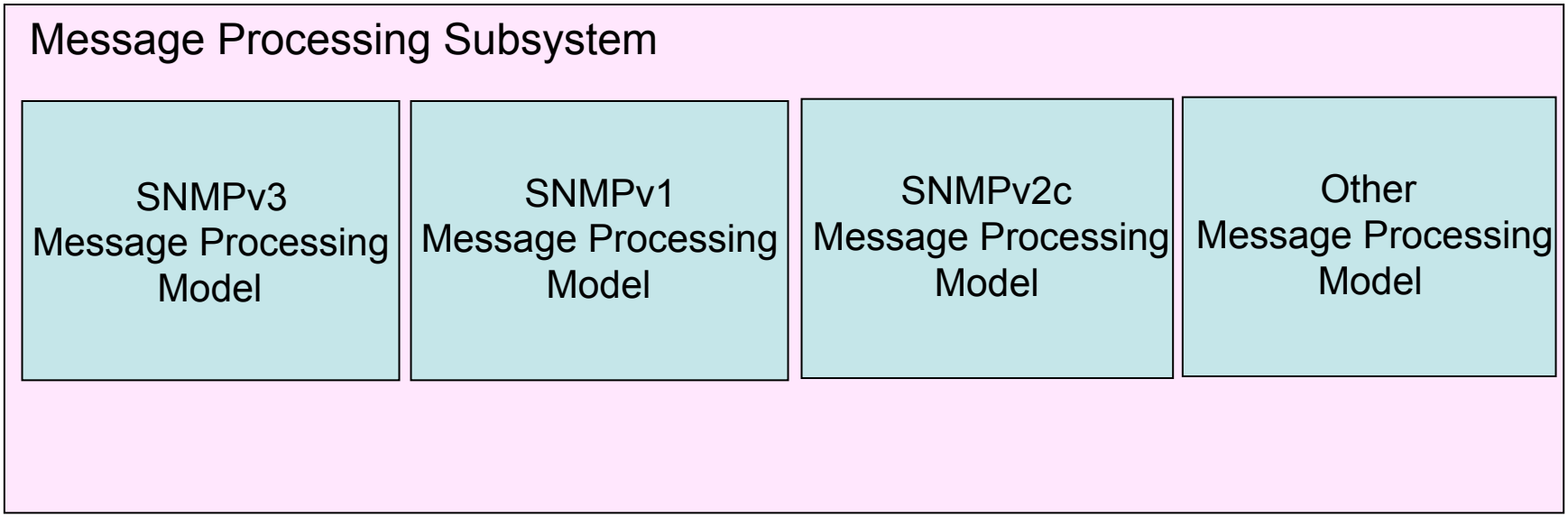
SNMP Engine - Dispatcher

- Responsible for sending and receiving SNMP messages to/from the network.
- Determines the version of an SNMP message and interacts with the corresponding Message Processing Model.
- Provides interface to SNMP applications:
  - for delivery of a PDU to an application
  - for the applications to send a PDU to a remote SNMP entity.

SNMP Engine – Message Processing Subsystem

- Responsible for:
  - preparing messages for sending, and
  - extracting data from received messages
- Contains multiple Message Processing Models
- Each Message Processing Model is specific to a particular version of an SNMP message and prepares and extracts version-specific message formats.

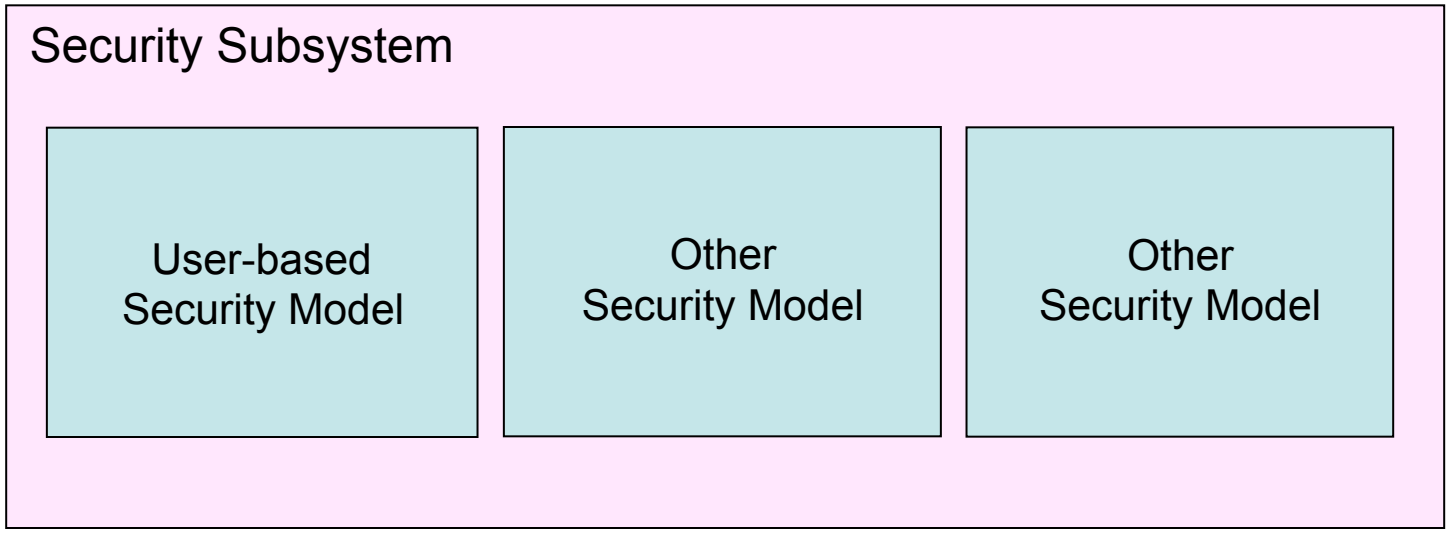
SNMP Engine – Message Processing Subsystem



SNMP Engine – Security Subsystem

- The Security Subsystem provides security services such as:
  - Authenticating messages, and
  - Encrypting/decrypting messages for privacy
- Contains one or more Security Models
- A **Security Model** specifies
  - the threats against which it protects, and
  - the security protocols used to provide security services such as authentication and privacy.

SNMP Engine – Security Subsystem



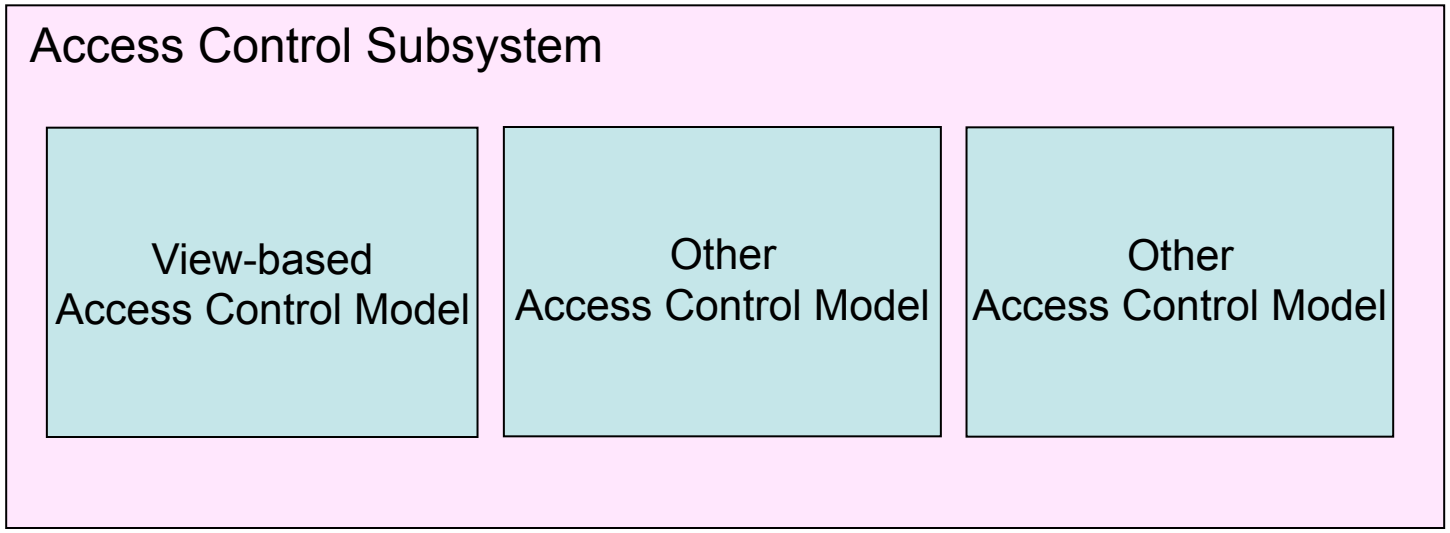
SNMPv3 Security Level

- Every message has an associated security level.
- SNMPv3 recognizes three levels of security:
  - without authentication and without privacy (noAuthNoPriv)
  - with authentication but without privacy (authNoPriv)
  - with authentication and with privacy (authPriv)

SNMP Engine – Access Control Subsystem

- The Access Control Subsystem provides authorization services by means of one or more Access Control Models.
- An **Access Control Model** defines a particular access decision function for computing access rights.

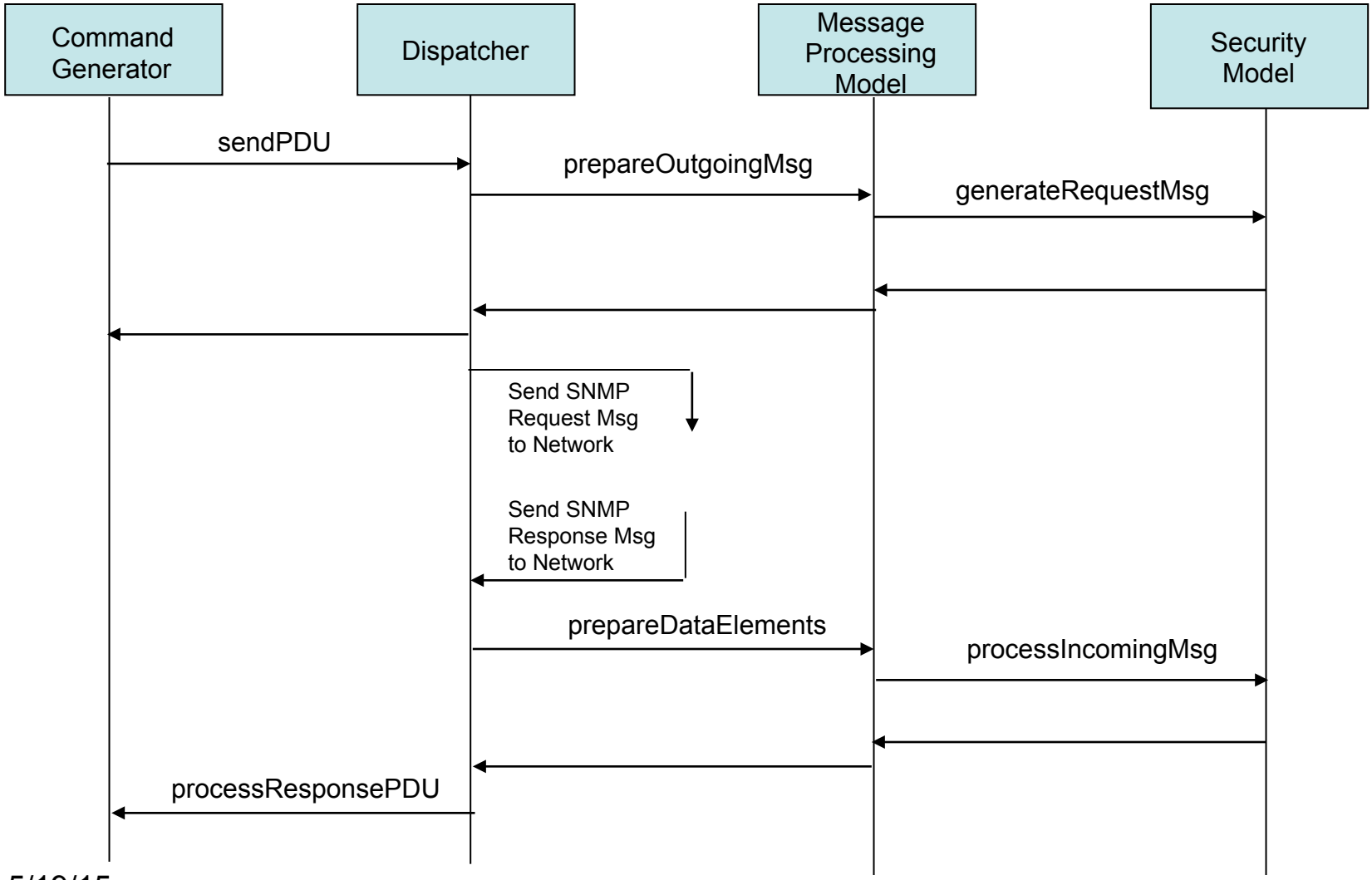
SNMP Engine – Access Control Subsystem



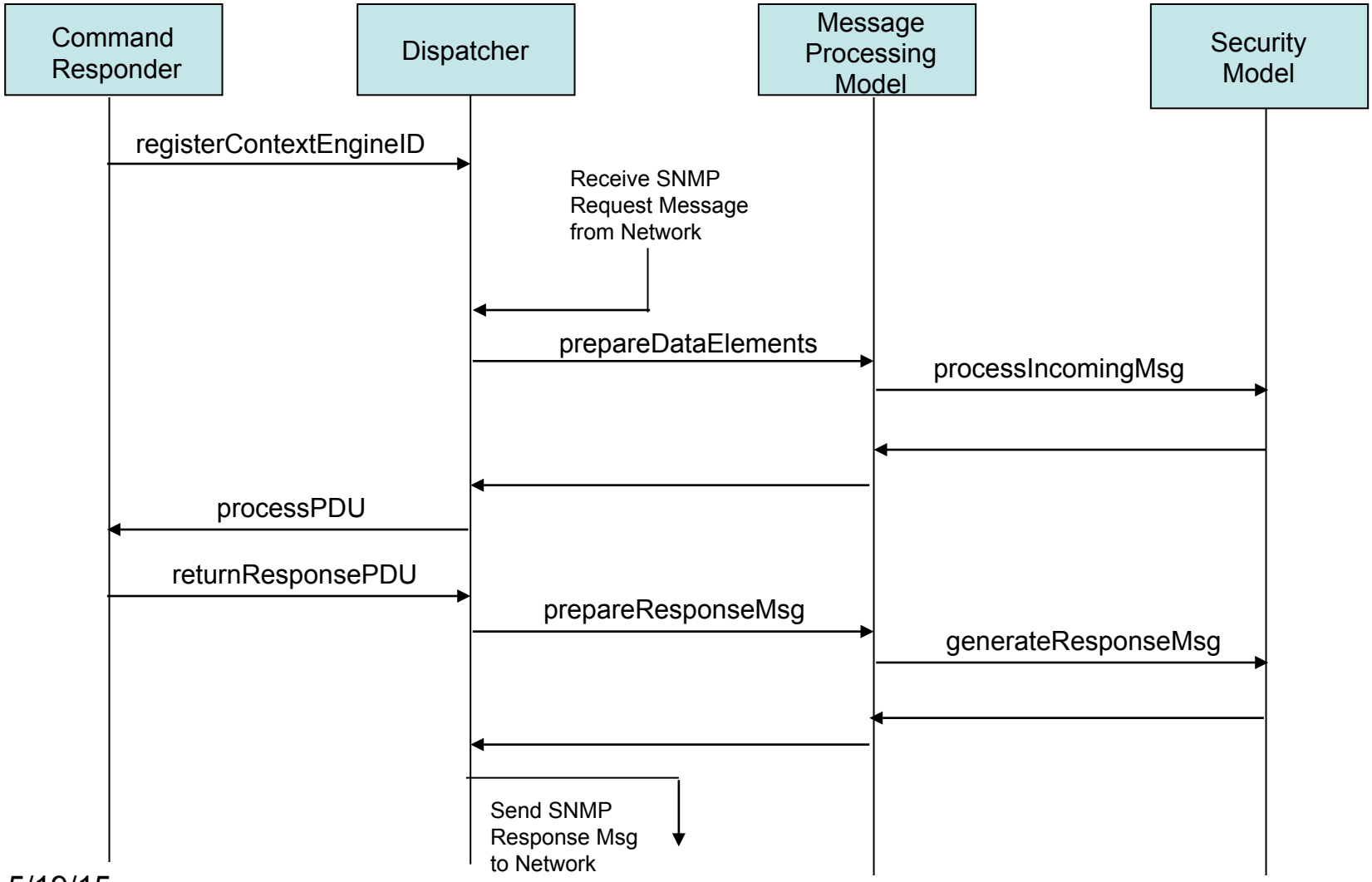
SNMP Applications

- SNMP applications make use of the services provided by the SNMP engine
- The different types of applications include:
  - **command generators**, which monitor and manipulate management data
  - **command responders**, which provide access to management data,
  - **notification originators**, which initiate asynchronous messages,
  - **notification receivers**, which process asynchronous messages, and
  - **proxy forwarders**, which forward messages between entities.

SNMPv3 Command Generator Call Flow



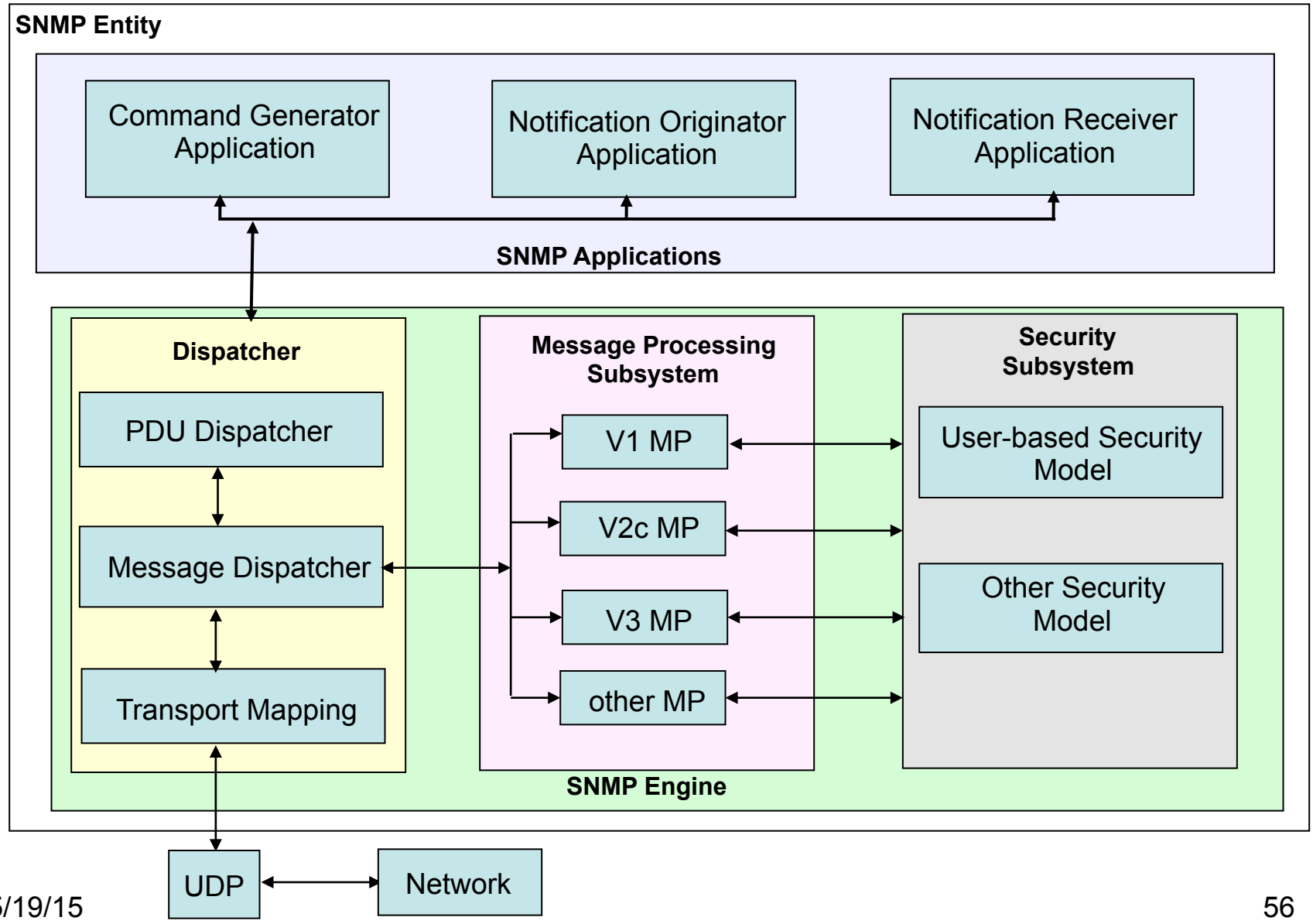
SNMPv3 Command Responder Call Flow



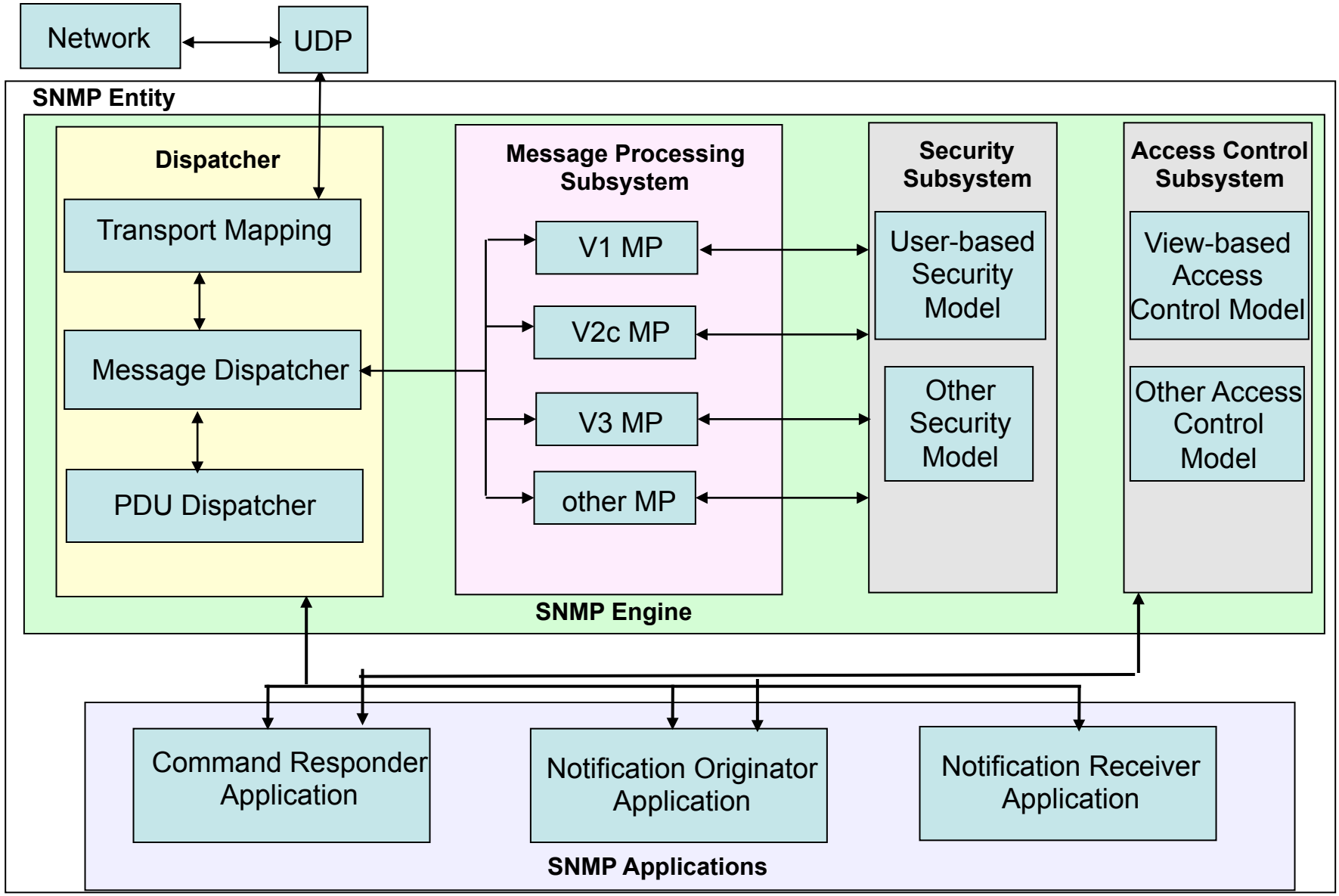
SNMPv3 Implementations

- Traditional SNMP agent: An SNMP entity with command responder and notification originator applications
- SNMP proxy agent: An SNMP entity with proxy forwarder application
- SNMP mid-level managers: entities with command generator and notification receiver, plus command responder and notification originator applications
- SNMP network management stations: entities with command generator, notification receiver and possibly other types of applications for managing a very large number of managed nodes

SNMP Manager



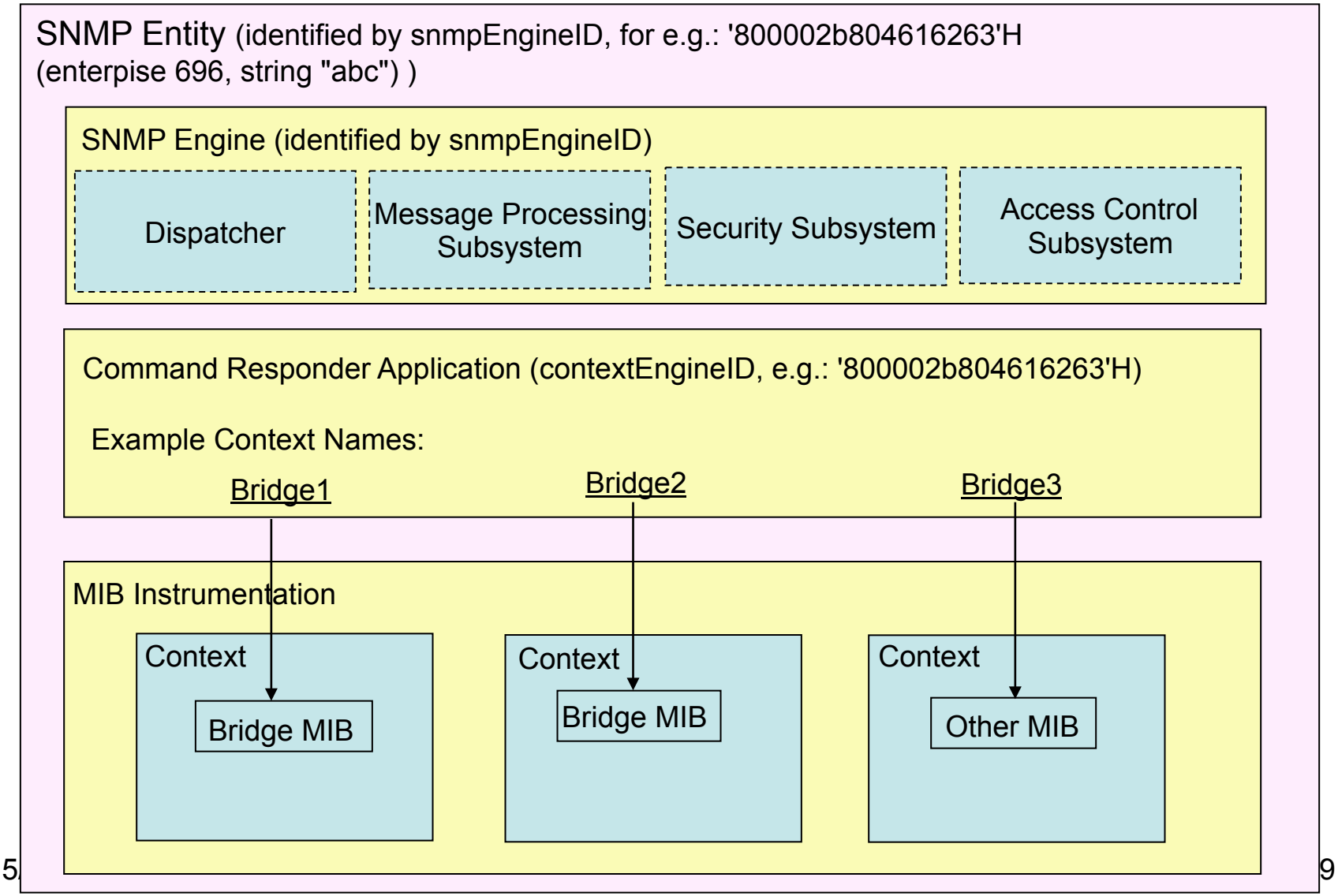
SNMPv3 Agent



SNMP Context

- An SNMP context is a collection of management information
  - An item of management information may exist in more than one context.
  - An SNMP entity potentially has access to many contexts.
- Often a context is a physical device or a logical device and is defined as a subset of a single SNMP entity.
- The combination of a contextEngineID and a contextName unambiguously identifies a context within an administrative domain
- To identify an individual item of management information, its contextName and contextEngineID must be identified in addition to its object type and its instance.

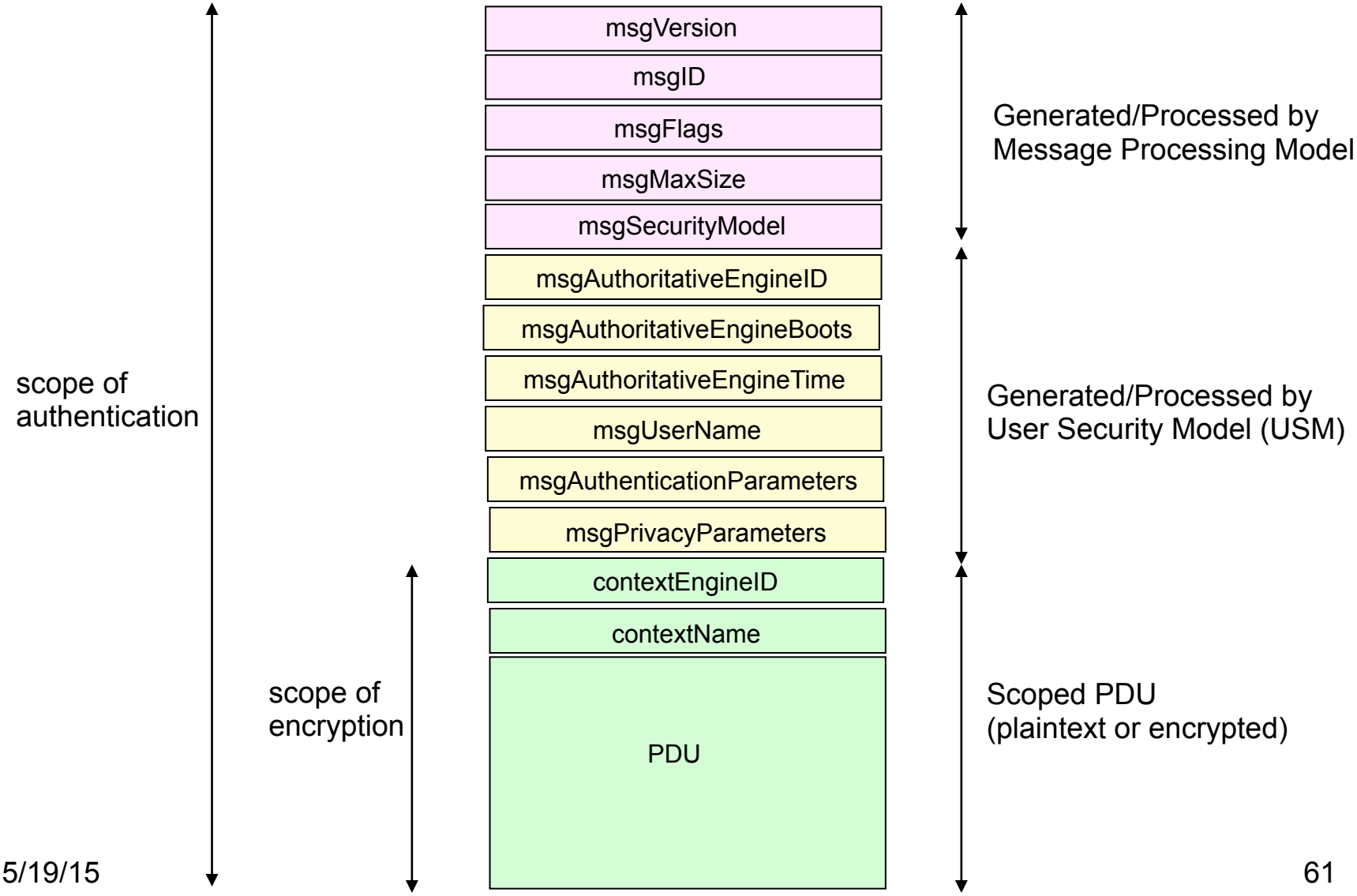
SNMP Context



SNMPv3 – Message Format

- Contains
  - Message Header
    - msgVersion
    - msgID
    - msgMaxSize
    - msgFlags
    - msgSecurityModel
  - Security Header
  - SNMPv2 PDU (plain text or encrypted)

SNMPv3 Message Format



SNMPv3 – Message Format

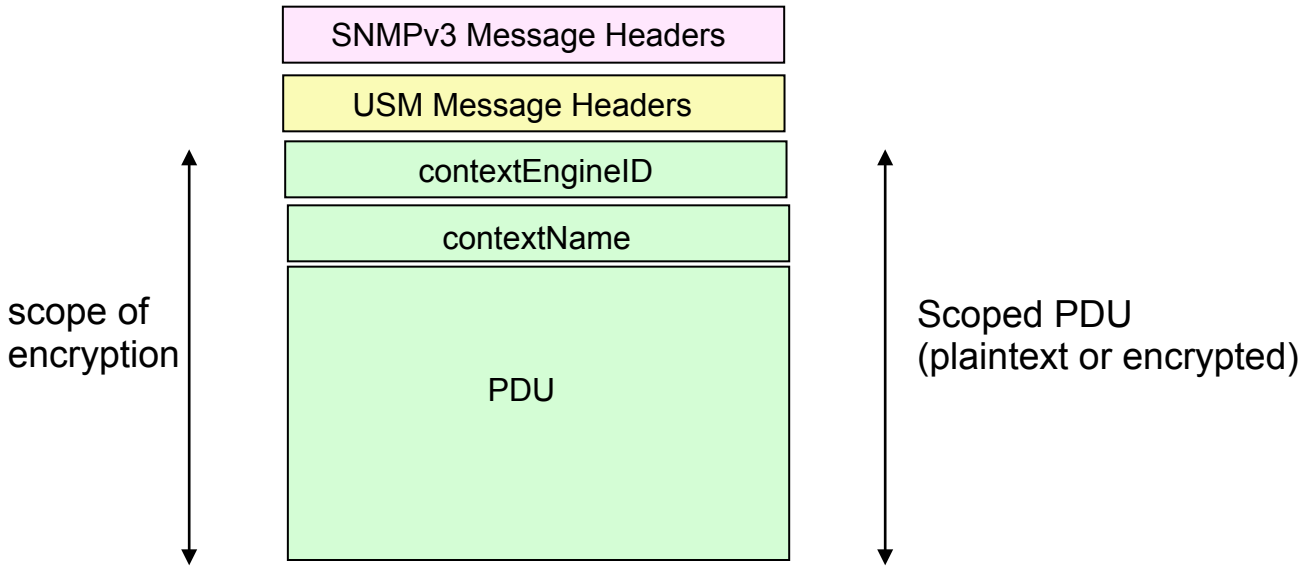
- msgVersion
  - A value of 3 identifies the version of the message as an SNMPv3 message
- msgID
  - Used to coordinate request and response messages between two SNMP entities
  - Similar to the reqID used within a PDU
- msgMaxSize
  - Indicates the maximum message size that the sender can support
  - Used to determine how big a response to a request message can be
  - Values range from 484 to 2^{31}-1

SNMPv3 – Message Format

- msgFlags
  - Indicates the security level that the sender has applied to the message.
  - The 3 bits defined are reportableFlag, authFlag, and the privFlag
- msgSecurityModel
  - An integer value which identifies the message security model that the sender used to generate the message.
  - The receiver must use the same security model to perform security processing for the message.

SNMP ScopedPDU

- A scopedPDU is a block of data containing a contextEngineID, a contextName, and a PDU.



SNMPv3 – Reports

- Used for engine-to-engine communication
- Processed by the SNMPv3 message processing model
- Allows to report the following type of errors (to the sender) when processing a SNMP message:
  - A response message could not be generated
  - A message was received with the authFlag cleared and the privFlag set
  - An error occurred while providing authentication and privacy services for an incoming message

SNMPv3 – Reports

type	reqid	0	0	Variable bindings

- The PDU type 0xA8 indicates a Report PDU
- ‘reqid’ consists of request identifier of the message that triggered the Report
- Variable bindings will contain a single object identifier and its value. Used to determine the problem that the report is identifying.