SNMP v1 - the protocol

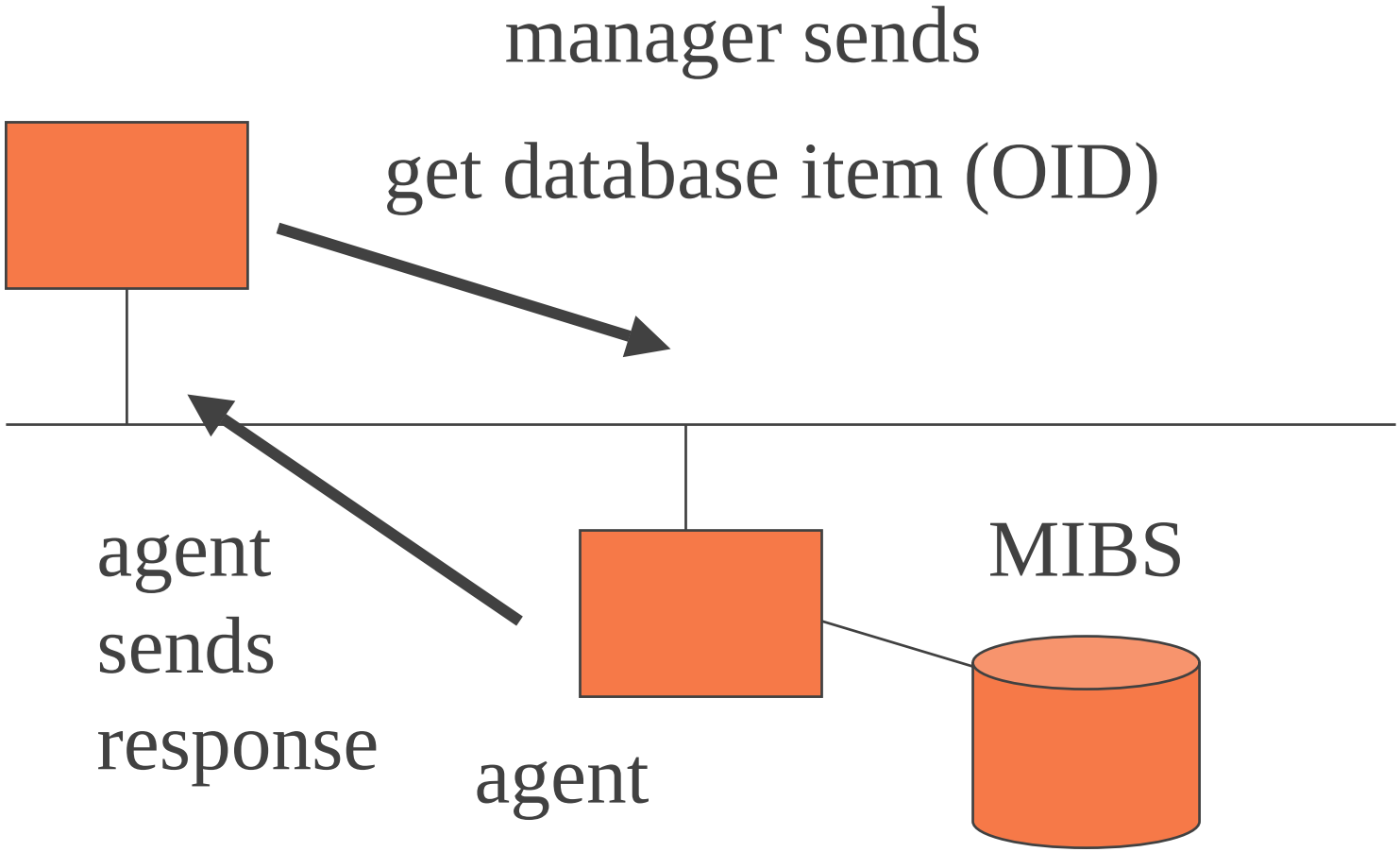
Network Mgmt/Sec.

Jim Binkley

Outline

- protocol overview
  - UDP/polling/reliability
- protocol formats
  - get, set, response, trap
  - get-next, and tables
- protocol examples

manager/agent



SNMP layered on top of UDP

- manager sends packet (snmp get) with:
  - ip src = manager, ip dst = agent
  - udp src port = client port (>=1024), say 1046
  - udp dst port = 161
- response formed by client comes back from port 161 to manager client port (1046)
- traps sent to port 162 on manager

UDP is unreliable protocol

- collisions, congestion, noise may kill packets
- send mechanism will typically have N retries
- UDP does have checksum mechanism (needs to be on)
- in addition, typical manager does periodic poll (5 minutes/15 minutes or longer)

polling, cont.

- in general you need to consider how many agents exist
- how many MIB objects are accessed (e.g., in BGP router, routing table could be quite large)
- how “far” in link counts is manager from worst-case agent
  - less links the better; manager should be central and should be on quality link

Heisenberg ...

- are you feeling “uncertain”?
- question: how can you measure capacity usage at manager?
- use HPOV on manager interfaces
- use MRTG to watch thruput on manager interfaces (or snmpget/snmpwalk)
- look at SNMP errors in snmp mib, interface errors in interface mib on manager

traps are fundamentally unreliable

- one way, agent to manager
- doesn’t mean they aren’t useful
- I rebooted (due to a crash)
- I lost an interface
- you get some redundancy here from syslog though
  - syslog is not SNMP ...

v1 community

- each agent may have 1-N community strings defined associated with access-mode
- access-mode: { read-only (RO), read-write)
- manager needs to know community in order for access to succeed
- more formally: **MIB view**: subset (or all) of objects in MIB
- possible that different communities may have different views (but rare)

community, etc.

- MIB object has ACCESS defined with it
- access-mode (set in global agent configuration) may/may not agree
- there aren't any surprises here really:
- you can't set an object with RO access mode.
- you can't write a read-only access
- thus all mib objects (if you are careful) can be read-only (not unusual due to paranoia)

community, etc

- **public** is often (always?) default for RO community
- **private** is often (not always) default for RW community
  - sometimes no default (good idea ...)
  - **check for this with any piece of equipment**
- typically multiple communities supported
  - may be ACL to limit access to certain IP addresses

naming (part 1)

- short-form:
- get (or other request) must utter the OID of the object
- basically: **snmp-get(1.3.6.1.etc = ???)**
- reply: **snmp-response(1.3.6.1.etc==value)**
- snmp messages like get can actually have multiple OID requests in them
- we will get to tables with snmp-get-next

packet (PDU) structure

- note PDU in pure SNMP speak refers to upstairs snmp type of packet
- SEQUENCE
  - version INTEGER {value is *0*)
  - community OCTET STRING,
  - data ANY -- actually the PDUs,, therefore

encapsulation

as an IP datagram

enet(MAC)	**ip (addr)**	**udp(port)**	**snmp (OID)**

common snmp structure layout

version	**community string**	**SNMP PDU/s**

note: remember BER, starts with sequence TLV

5 kinds of PDU (packet)

- PDUs ::= CHOICE
  - get-request GetRequest-PDU,
  - get-next-request GetNextRequest-PDU,
  - get-response GetResponse-PDU,
  - set-request SetRequest-PDU,
  - trap Trap-PDU

common PDU structure

- for all but Trap-PDU
- Get (0), GetNext (1), GetResponse(2), Set (3)
- PDU ::= SEQUENCE
  - request-id INTEGER
  - error-status INTEGER
  - error-index INTEGER
  - variable binding VarBindList
- note: > 1 variable bindings
- varBind is: (name ObjectName, value ObjectSyntax) (NULL in gets)

notes (for all but trap)

- error-status values include:
  - noError (0)
  - tooBig (1) - GetResponse PDU < MIB value
  - noSuchName (2) - no known OID in MIB/s
  - badValue (3) - setRequest has “bad” value
  - readOnly (4) --
  - genError(5) -- general error, see error-index
- error-index, if > 0, index into variables in variable bindings (which one is wrong)

encapsulation of gets/set/request

get, get-next, set

PDU type	request-id	0	0	(name,value) (name,value)

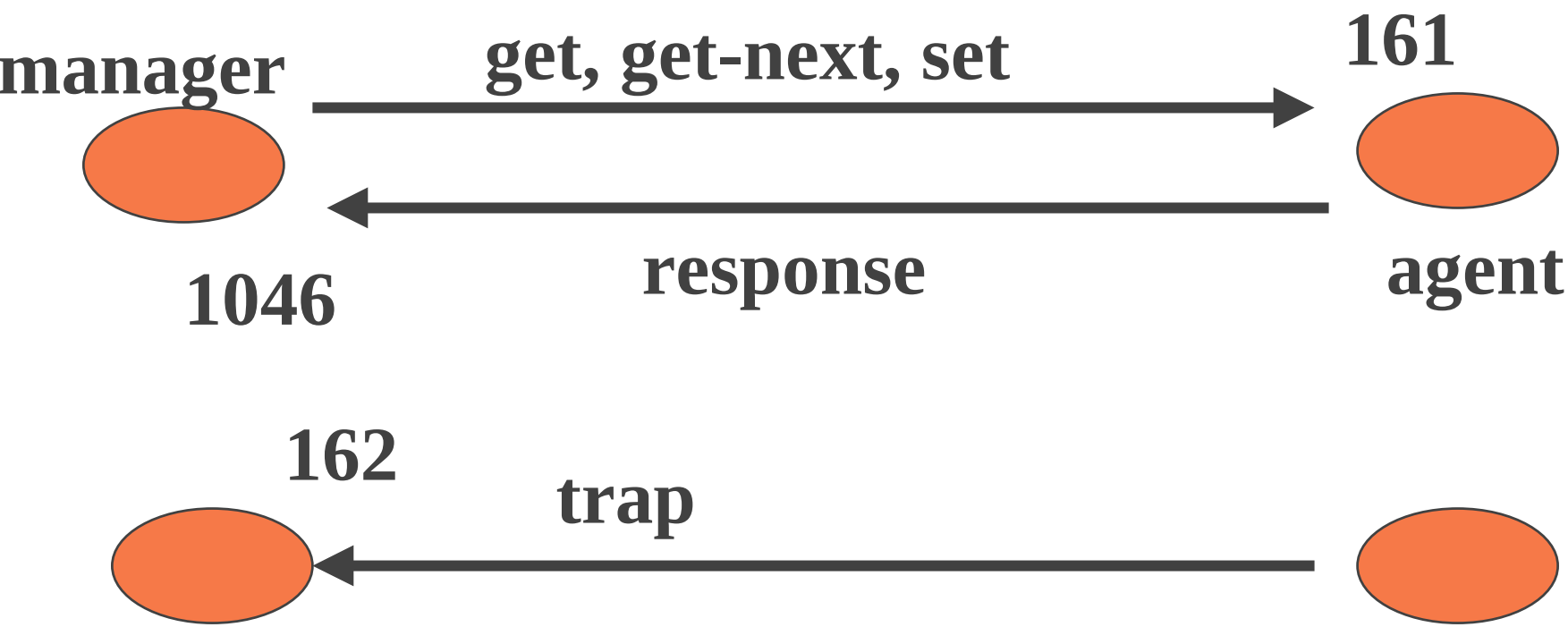
get-response

PDU type	request-id	error-status	error-index	vars

notes

- typically one, but > 1 in variable bindings possible
- very unlikely but one UDP packet can contain multiple snmp messages ...
- error fields are 0 in gets/set, possibly non zero in response
- request-id in get and response should be same - effectively sequence number

protocol picture



snmp-get - 1 MIB variable

ucd-snmpget example #1:

snmpget -d -v 1 DNS-name COMMUNITY system.sysDescr.0

sending 49 bytes to 131.252.222.201:
30 82 00 2D **02 01 00** 04 06 70 75 62 6C 69 63 A0 0.....public
20 02 04 3B F4 06 EB 02 01 00 02 01 00 30 82 00 ..;t.k.....0..
10 30 82 00 0C 06 08 2B 06 01 02 01 01 01 00 05 .0.....+.....
00 .

note: done on localhost, above is snmp-get

snmp-response part

received 178 bytes from 131.252.222.201
30 82 00 AE 02 01 00 04 06 70 75 62 6C 69 63 A2 0.....public"
82 00 9F 02 04 3B F4 06 EB 02 01 00 02 01 00 30 .....;t.k.....0
82 00 8F 30 82 00 8B 06 08 2B 06 01 02 01 01 01 ...0.....+.....
00 04 7F 46 72 65 65 42 53 44 20 7A 79 6D 75 72 ...FreeBSD etc.

... truncated, following is displayed response for **sysDescr**

system.sysDescr.0 = "FreeBSD mumble.cs.pdx.edu
 2.2.1-RELEASE FreeBSD 2.2.1-RELEASE #2:
 Mon Jul 14 09:12:23 PDT 1997   trost@billboy.cs.pdx.edu:/v"

tcpdump of snmp-get

```
# tcpdump -i lo0 udp and port 161

14:40:08.177705 131.252.222.201.3739 > 131.252.222.201.161:
 |30|82|00|2d|02|01|04|06|a0|20GetRequest(19)|02|04|02|01|02|01|30|
82|00|10 |30|82[|snmp] (ttl 64, id 32901)

                    data deleted ...
```

tcpdump of same response

14:40:08.177895 131.252.222.201.161 > 131.252.222.201.3739:
|30|82|00|ae|02|01|04|06|a2|82|00|9fGetResponse(17)|02|04|02|01|02|
01|30|82|00|8f |30[|snmp] (ttl 64, id 32902)

data deleted ...

get-next and tables

- two kinds of access in SNMP
- 1. “random access”, which simply means you supply the **exact OID**, and get/set value
  - tricky if table entry, else simple
- 2. “sequential access”, which is done on basis of **lexicographical ordering** of OIDs

lexicographical ordering

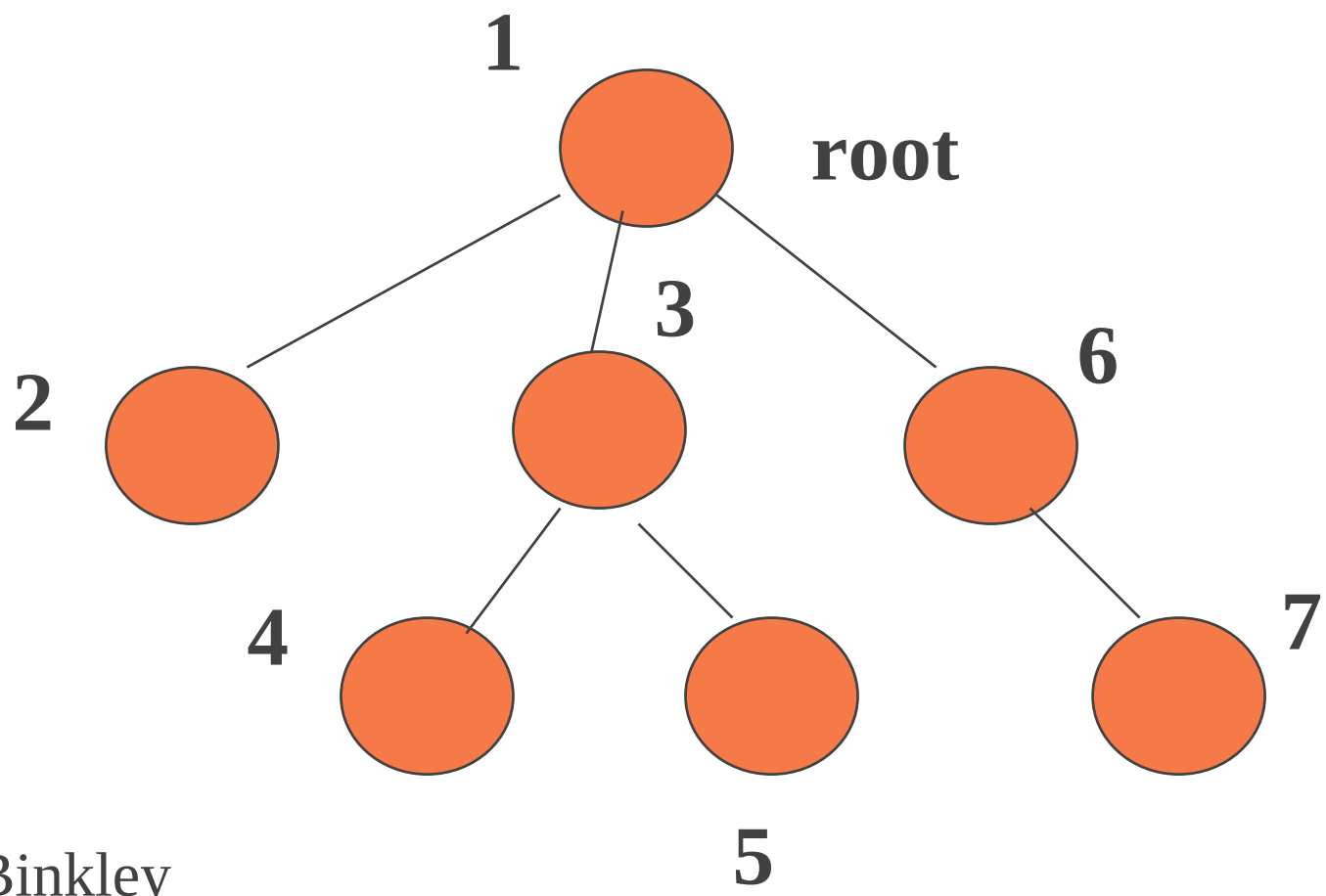
- an OID is a series of small integers from left to right (x1, x2, ... xn)
- we say that OID1 precedes OID2 if for (x1, x2, ... xn) and (y1, y2, ... y2)

  OID1 < OID2 if

  iterate via index i, and while xi = yi

  stop if xi is < yi

put another way



lexicographic order means:

- visit the root and then traverse subtrees from left to right
- depth-first search of the tree
- so: snmp-get-next given OIDx
- returns OID(next lexicographic leaf)
- basically works on leafs
- but given non-leaf predecessor, finds next leaf

lexicographic order

- is how managers can walk tables without a priori knowing how big table is
- e.g., IP MIB-2 section contains:
  - routing table
  - arp table
- could be arbitrarily big (and change periodicially)

tables, like Gaul

- have three parts
  - 1. table name (ipNetToMediaTable(22))
  - 2. row name (ipNetToMediaEntry(1)
  - 3. column object name (ipNetToMediaIfIndex(1))
- table
  - row name
    - » column name

hypothetical example

	name	number	age
r1	billy	725-1111	100
r2	sue	725-1212	1
r3	bob	725-1234	54
	c1	**c2**	**c3**

in snmp speak:

- this is the table peopleTable, sequence of
- made up of rows peopleRows, sequence
- name, number, age (column names)
- random access in snmp would be: peopleTable.peopleRow.number.725-1212
- i.e., r2, c2
- assumption: index is put at end, and somehow uniquely identifies item

therefore

- OID for tables of form: table.row.column-label.index
- supplied to get-next, returns next table entry (or next entry period -- can walk off table)
- next table entry COLUMN-WISE
  - NOT ROW-WISE
- random access requires index at end for table

reality-check

- there aren't that many tables in MIB-2
- in general index is straight-forwarded
- not so, for tcp connection table
  - index is 4 tuple of ip peers addresses and tcp client/server ports
- easier to use get-next and walk through table than utter "random-access" name
- e.g., ucd-snmp snmpwalk > snmpget

ARP table - ASN (roughly)

- ip.ipNetToMediaTable (table name)
  - ipNetToMediaEntry (row name)
    - » index is: (ipNetToMediaIfIndex, NetAddress)
    - » SEQUENCE
      ipNetToMediaIfIndex INTEGER,
      ipNetToMediaPhysAddress PhysAddress (MAC)
      ipNetToMediaNetAddress IpAddress
      ipNetToMediaType INTEGER
- basically (MAC, IP) bound to a certain interface (ifIndex)

handout time

- look at handout for arp table walk

traps

- Trap-PDU ::= [4] IMPLICIT SEQUENCE
  - enterprise OBJECT IDENTIFIER -- system.sysObjectID - equipment ID
  - agent-addr NetworkAddress -- ip address
  - generic-trap INTEGER -- type of trap
  - specific-trap INTEGER
  - time-stamp TimeTicks -- time since boot
  - variable-bindings -- interesting info

trap types

- **coldStart** (0) -- reboot possibly due to crash
- warmStart (1) -- software reinit
- **linkDown**(2) -- link crashed
- linkUp(3) -- link uncrashed
- authenticationFailure(4) -- invalid SNMP password
- egpNeighborLoss(5)
- enterpriseSpecific(6)

what to do with traps

- besides log (e.g., ucd snmptrapd can log in UNIX syslog)
- treat as 1st-class faults and
- send linkDown coldStart with associated strings to pager
- note Cisco enterprise traps can tell you about things like router re-configuration (which could be a security event ...)

snmp trap - cisco cold start

on cisco# **reload**

reboot trap send to manager ...
Mar 29 17:38:21 deedee **snmptrapd**[2580]: 131.252.215.4: Cold Start Trap (0) Uptime: 0:00:08, system.sysUpTime.0 = Timeticks: (858) 0:00:08.58, enterprises.9.2.1.2.0 = "reload”

note: ucd-snmp unix snmptrapd logs to syslog

snmp trap - cisco link down

cisco-config-interface# interface FastEthernet0/1
shutdown

Mar 29 17:39:59 deedee snmptrapd[2580]: 131.252.215.4: Link Down Trap (0) Uptime: 0:02:10,
interfaces.ifTable.ifEntry.ifIndex.2 = 2,
interfaces.ifTable.ifEntry.ifDescr.2 = FastEthernet0/1,
interfaces.ifTable.ifEntry.ifType.2 = ethernetCsmacd(6),
enterprises.9.2.2.1.1.20.2 = "administratively down”

note: how can figure out what MIB was used for the enterprises MIB object? what other MIB objects are in that MIB?

enterprise trap/reality check?!

note: this showed up. what is it and can we make it more meaningful?

Mar 29 17:40:56 deedee snmptrapd[2580]: 131.252.215.4: Enterprise Specific Trap (1) Uptime: 0:03:20, enterprises.9.9.43.1.1.6.1.3.4 = 1, enterprises.9.9.43.1.1.6.1.4.4 = 3, enterprises.9.9.43.1.1.6.1.5.4 = 2

example v1 setup/cisco router

- in IOS speak:
- access-list 2 permit <manager-ip-address>
- snmp-server community MYNET RO 2
- snmp-server community MYNETFOO RW 2
- snmp-server host manager-ip <COMM> snmp config
- snmp-server location under the bed in my room
- snmp-server contact anybody but me, x-1234
- # show snmp (to see setup)

cisco setup notes

- don’t do RW if you don’t believe in “writeability” or don’t have to
  - ancient security principle: **least privilege** - don’t give it up unless you have to ...
- set the location/contact
  - for the sake of the person covering for you on vacation ...
- cisco enterprise traps may be specified (or not) in the trap line