# SNMP Trace Analysis Definitions

draft-schoenw-nmrg-snmp-trace-definitions-02.txt

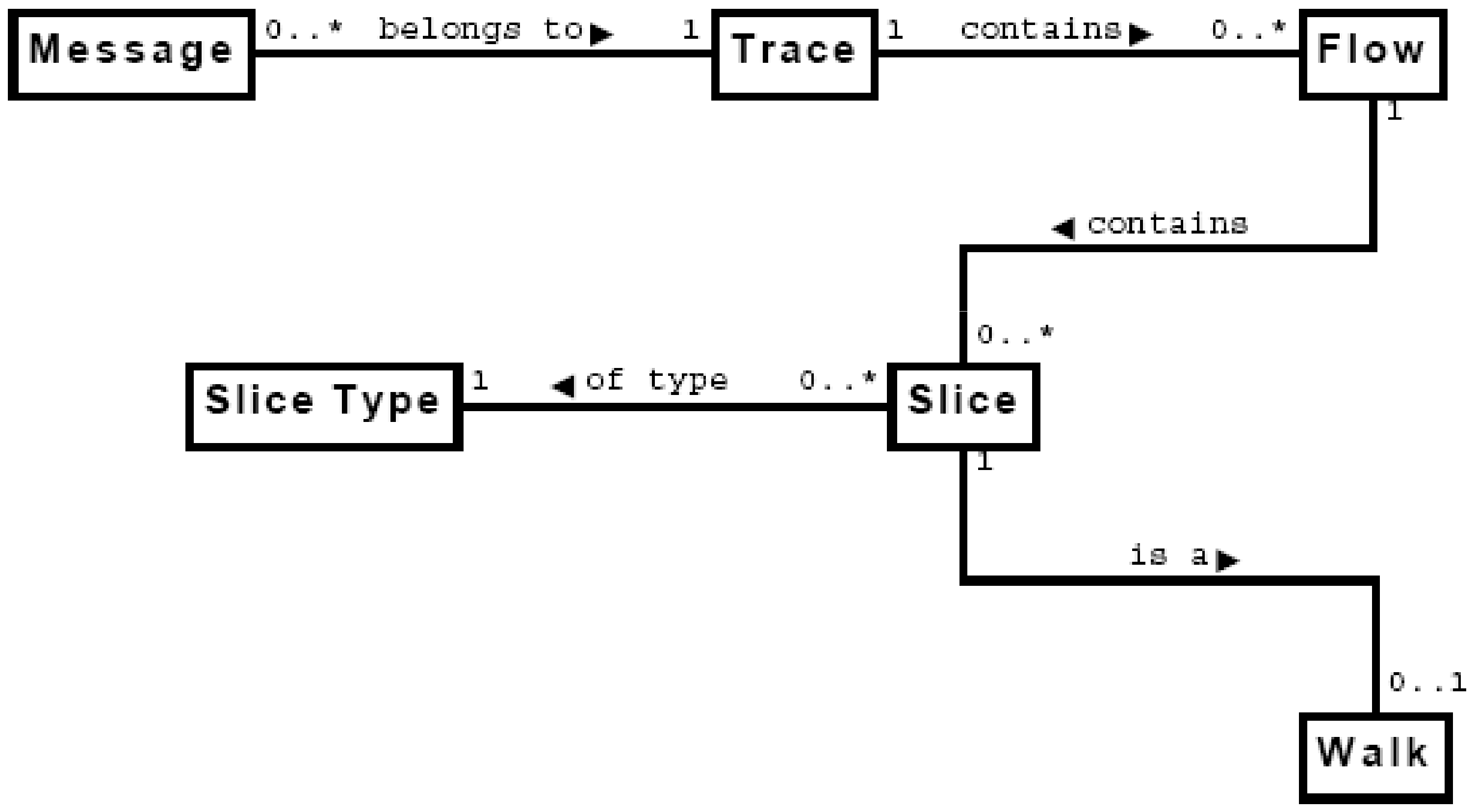
J.G. van den Broek
J. Schoenwaelder
A. Pras
M. Harvan

## We want feedback

## Why this draft?

- Extensive use of SNMP has led to significant practical experience
- Availability of SNMP traces
- Research indicates relations between SNMP messages
- Goal: common set of definitions between the researching parties

## Overview


Message
0..*
belongs to ▶
1
Trace
1
contains ▶
0..*
Flow
1
◀ contains
0..*
Slice Type
1
◀ of type
0..*
Slice
1
is a ▶
0..1
Walk

## SNMP Messages

- Definition (**read request message**): A read request message is a message M containing a PDU of type *GetRequest*, *GetNextRequest*, or *GetBulkRequest*.

- Definition (**write request message**): A write request message is a message M containing a PDU of type *SetRequest*.

- Definition (**notification request message**): A notification request message is a message M containing a PDU of type *InformRequest*.

- Definition (**request message**): A request message is a message M which is either a read request message, a write request message, or a notification request message.

# SNMP Messages

- Definition (**response message**): A response message is a message M containing a PDU of type *Response* or of type *Report*.
- Definition (**non-response message**): A non-response message is a message M which is either a read request message, a write request message, or a notification message.

# SNMP Messages

- Definition (**command message**): A command message is a message M which is either a *read request message* or a *write request message*.

- Definition (**command group messages**): A set of command group messages consists of all messages M satisfying either of the following two conditions:

(C1) M is a command message

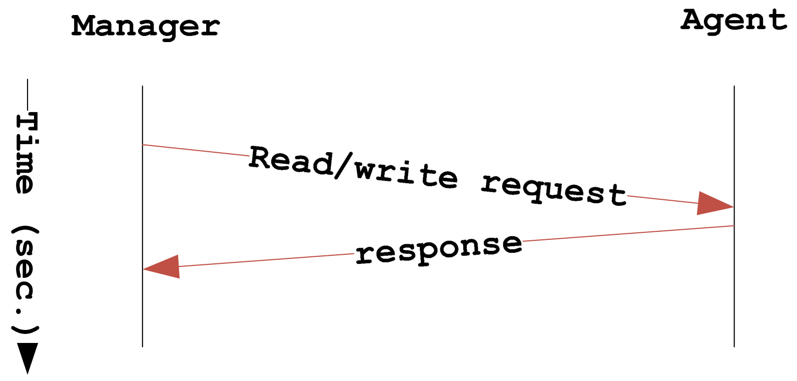
(C2) M is a response message and there exists a command message C such that the following holds:

M.reqid = C.reqid

M.tdst = C.tsrc

M.tsrc = C.tdst

(M.time - C.time) < t



The parameter t defines a maximum timeout for response messages.

## SNMP Messages

- Definition (**notification message**): A notification message is a message M containing a PDU of type *Trap* or *InformRequest*.

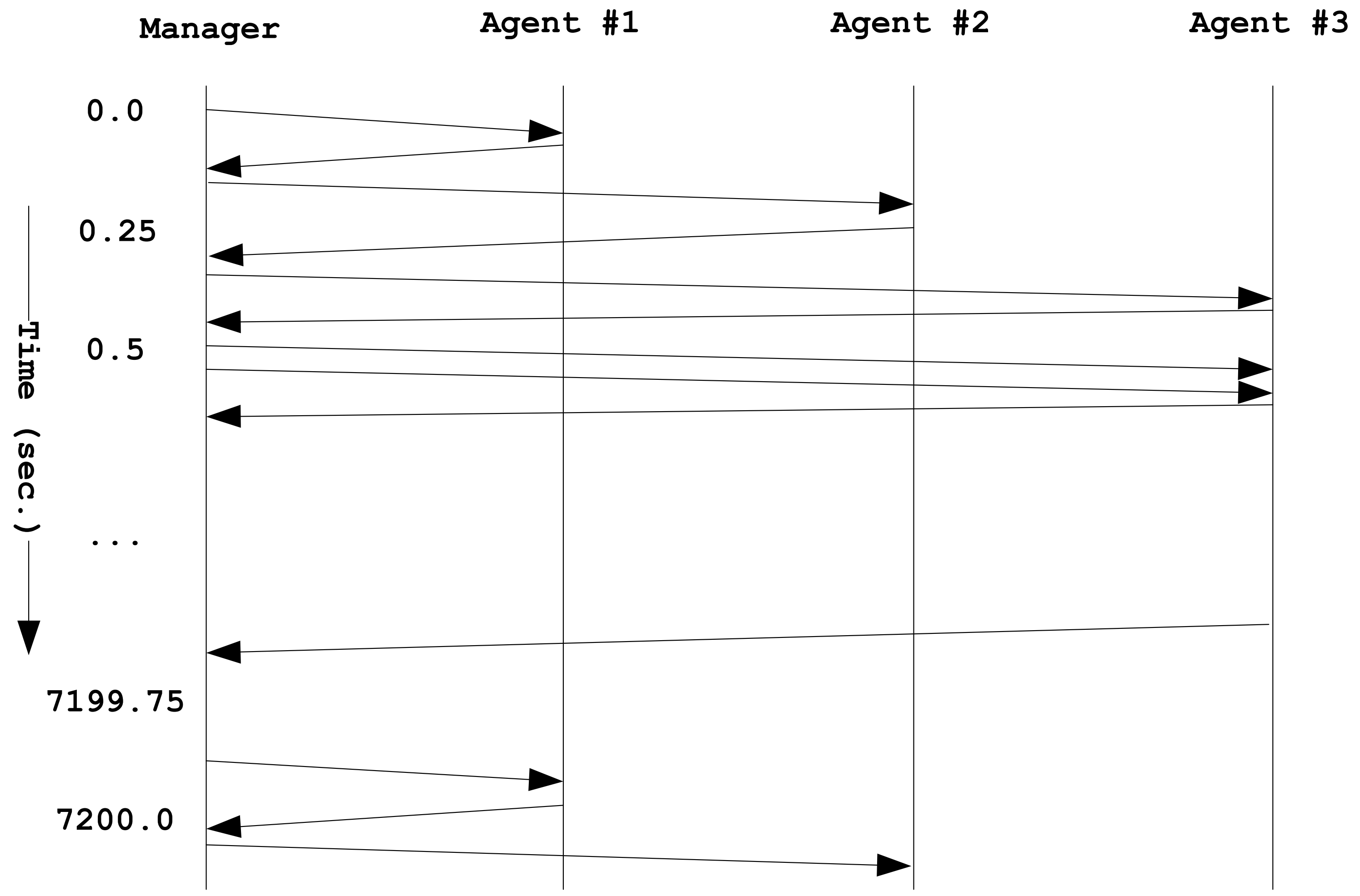
- Definition (**notification group messages**): A set of notification group messages consists of all messages M satisfying either of the following two conditions:

  (N1) M is a notification message
  (N2) M is a response message and there exists a notification request message N such that the following holds:

  M.reqid = N.reqid
  M.tdst = N.tsrc
  M.tsrc = N.tdst
  (M.time - N.time) < t

  The parameter t defines a maximum timeout for response messages.

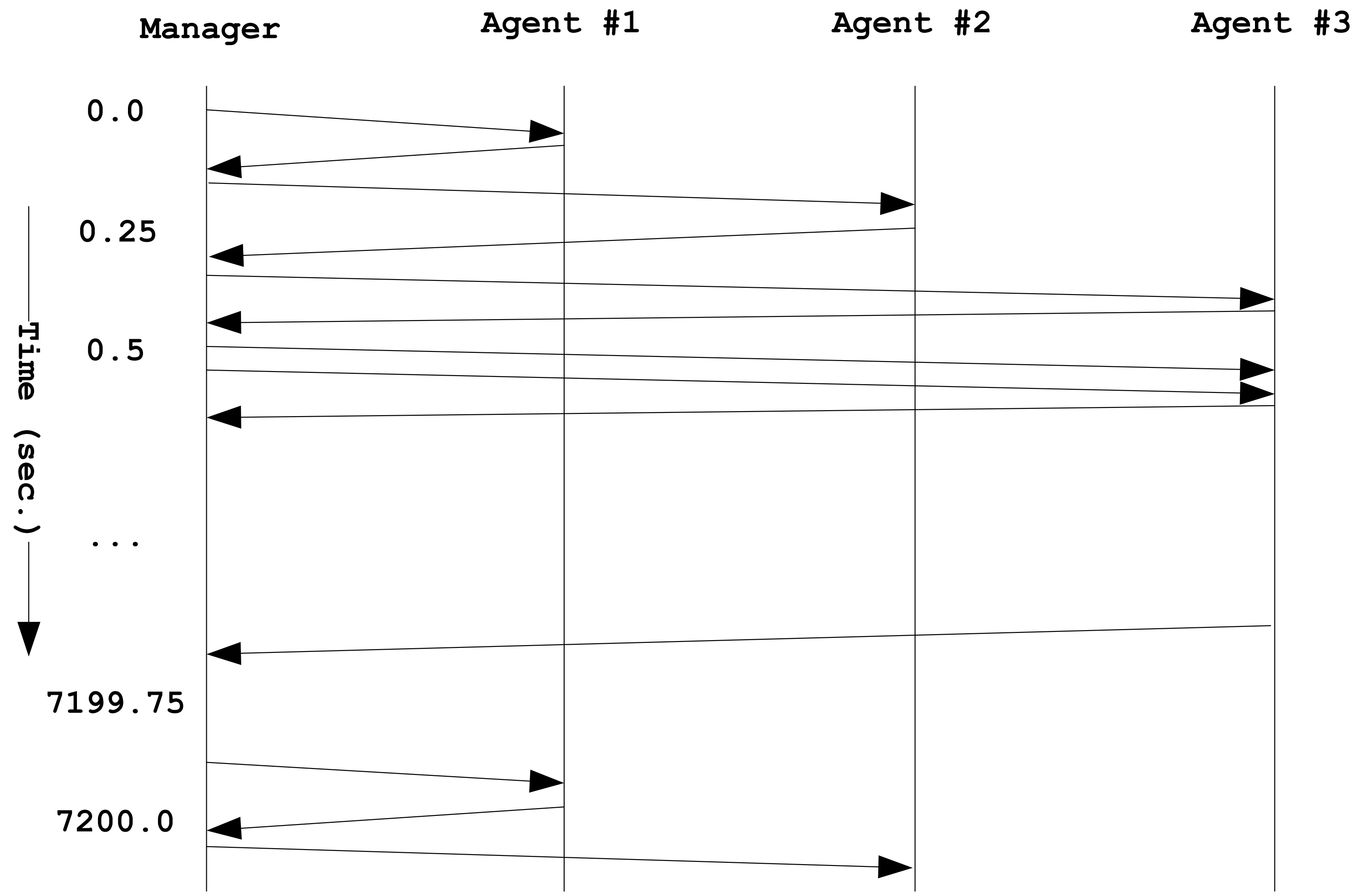
# Traces


Manager
Agent #1
Agent #2
Agent #3
0.0
0.25
0.5
...
7199.75
7200.0
Time (sec.)

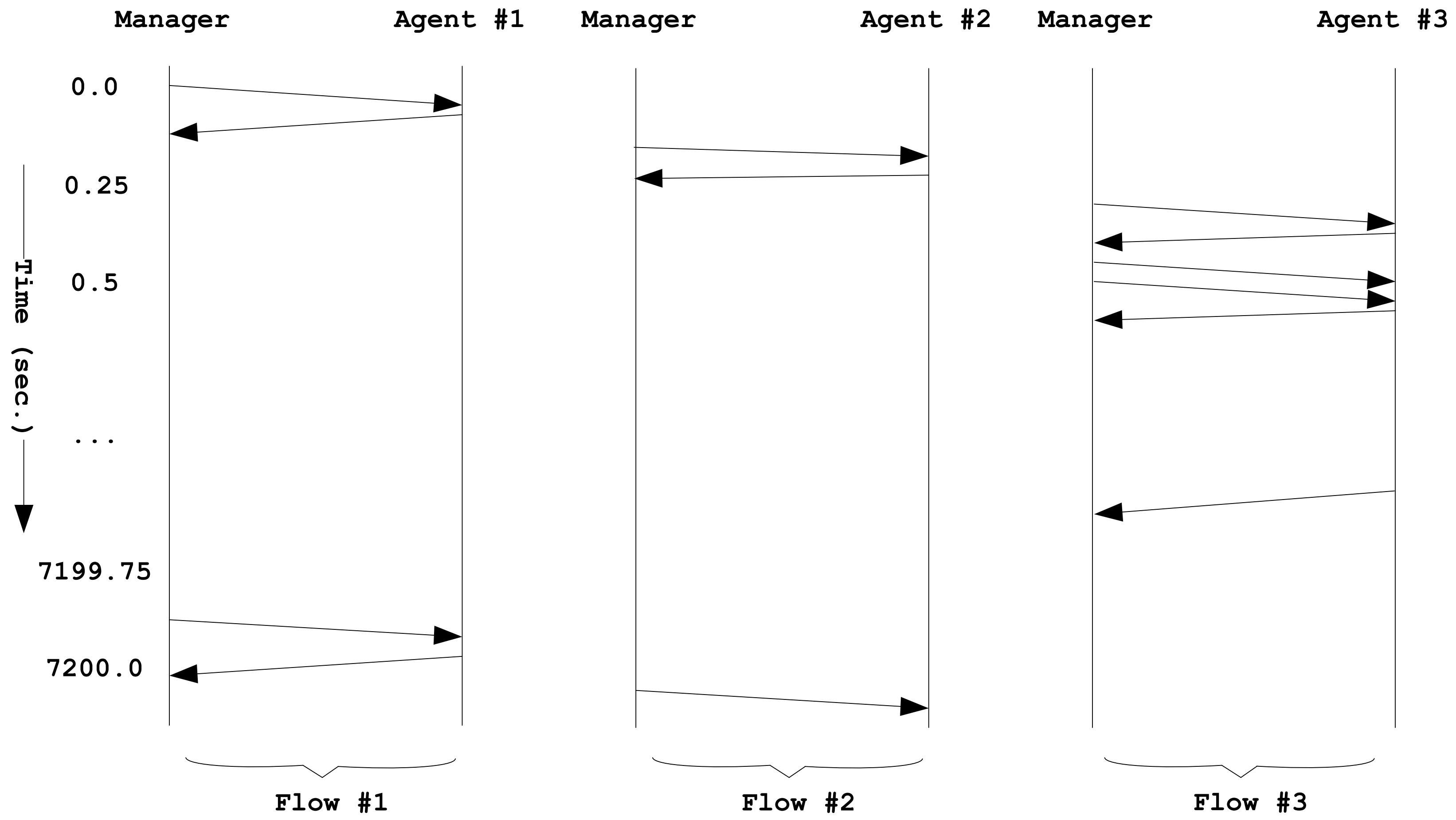
## Traces

- Definition (**trace**): An SNMP trace (or short "trace") T is an ordered set of zero or more SNMP messages M. All messages M in T are chronologically ordered according to the capture time stamp M.time.

# Flow


Manager
Agent #1
Agent #2
Agent #3
0.0
0.25
0.5
...
7199.75
7200.0
Time (sec.)

# Flow


Manager
Agent #1
Manager
Agent #2
Manager
Agent #3
0.0
0.25
0.5
...
7199.75
7200.0
Time (sec.)
Flow #1
Flow #2
Flow #3

# Flow

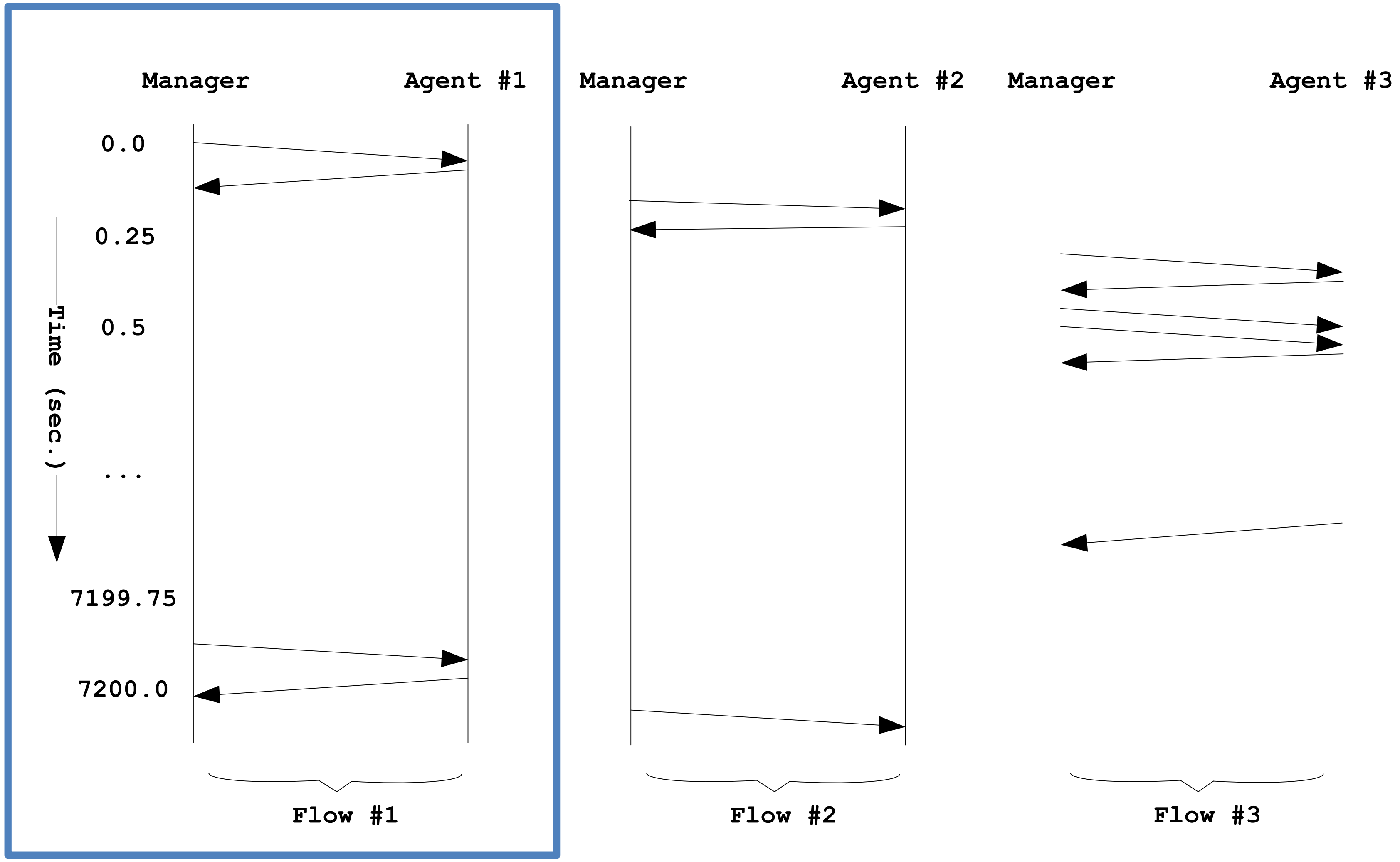
- Definition (**flow**): A flow F with parameter t is the set of messages of an SNMP trace T with the following properties:

  (F1) All response messages originate from a single network endpoint.

  (F2) All non-response messages originate from a single network endpoint.

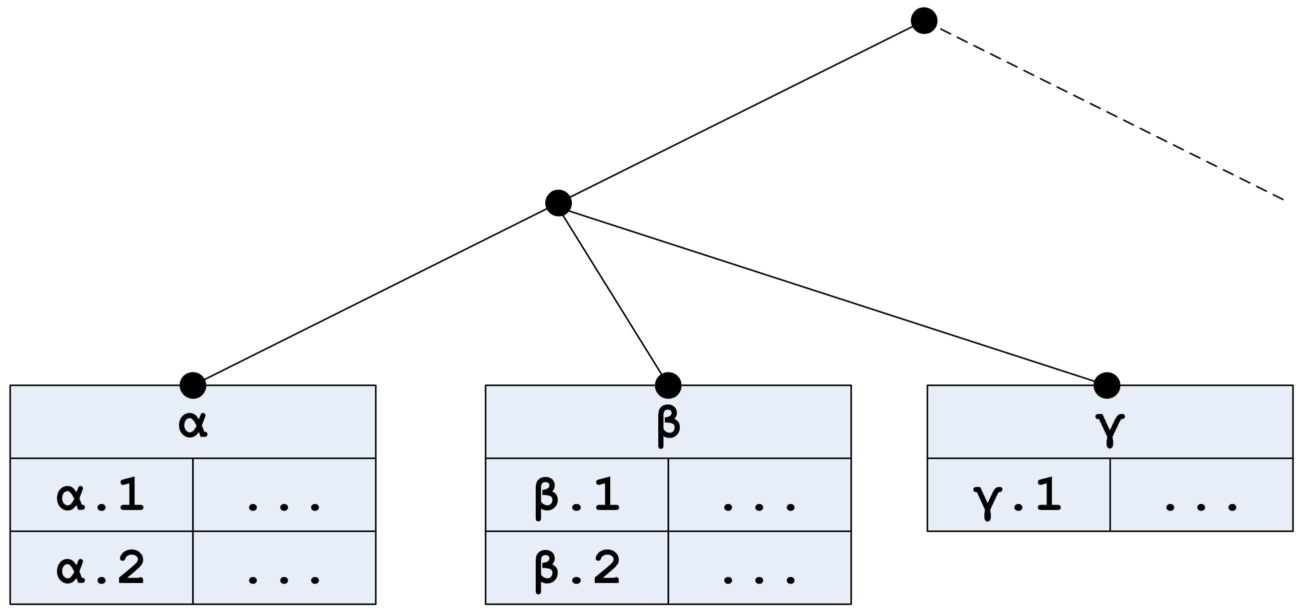
  (F3) All messages are either command group messages with parameter t or notification group messages with parameter t.

# Flows

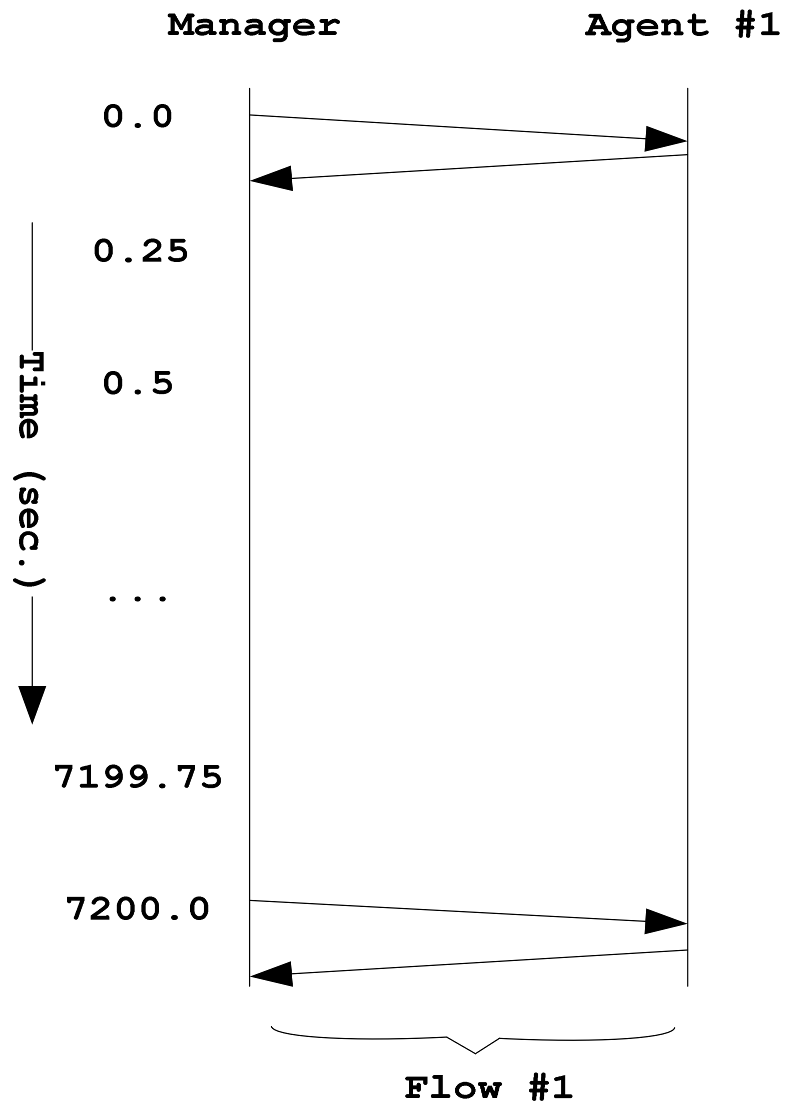

Manager
Agent #1
Manager
Agent #2
Manager
Agent #3
0.0
0.25
0.5
...
7199.75
7200.0
Time (sec.)
Flow #1
Flow #2
Flow #3

# Agent #1 characteristics

- Consider agent #1 with the following available tables:

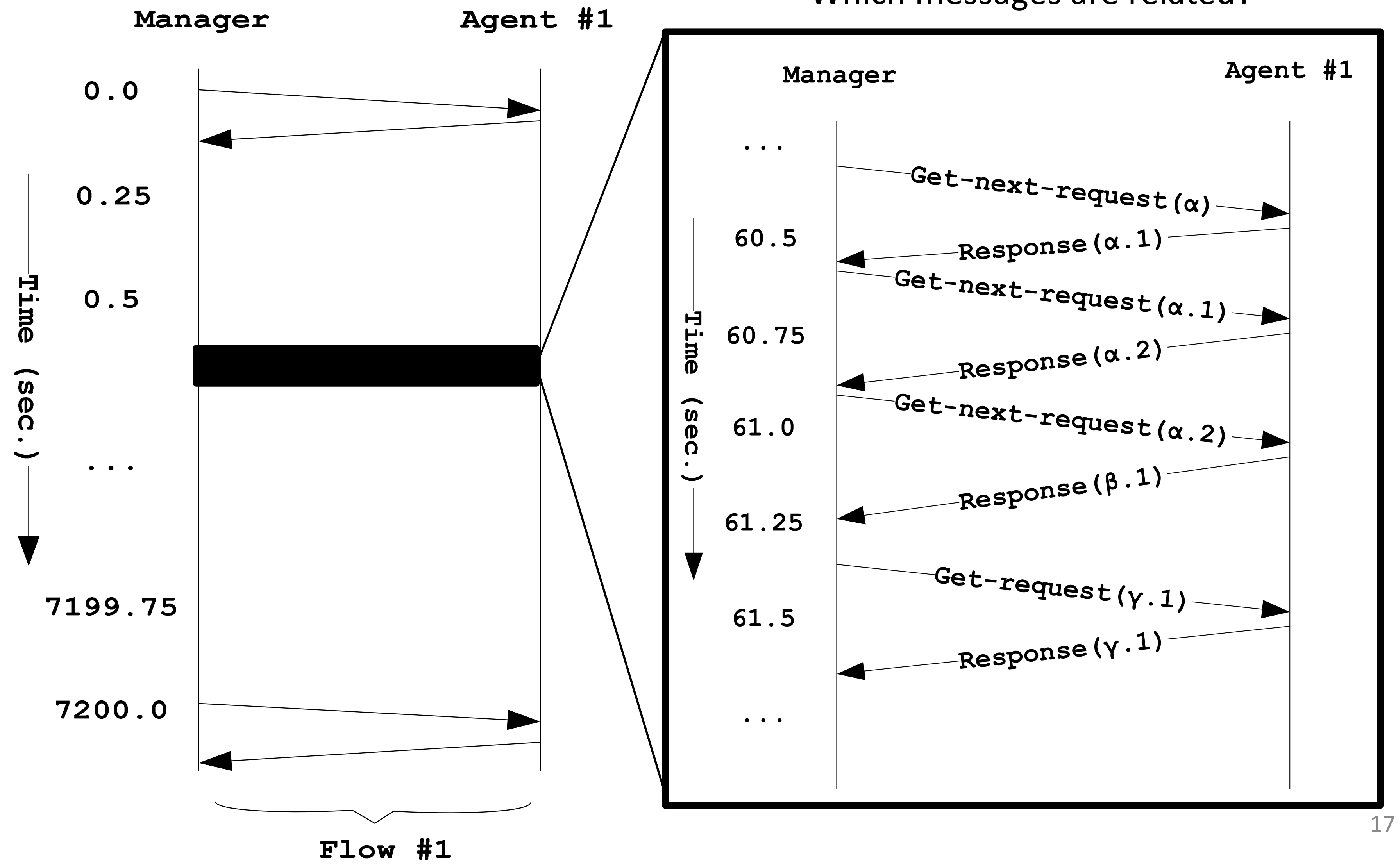


# Part of a single flow



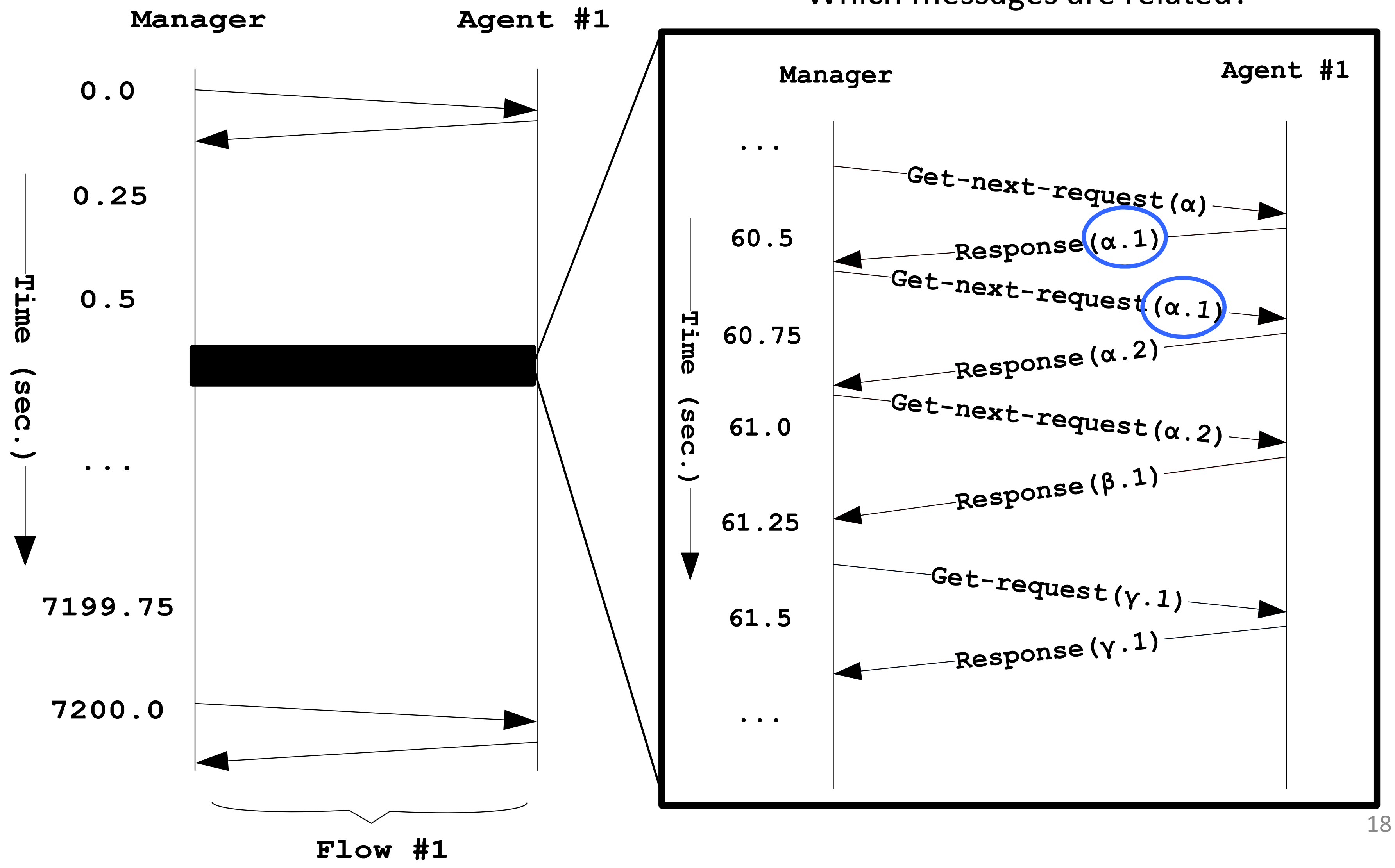
# Part of a single flow

Which messages are related?



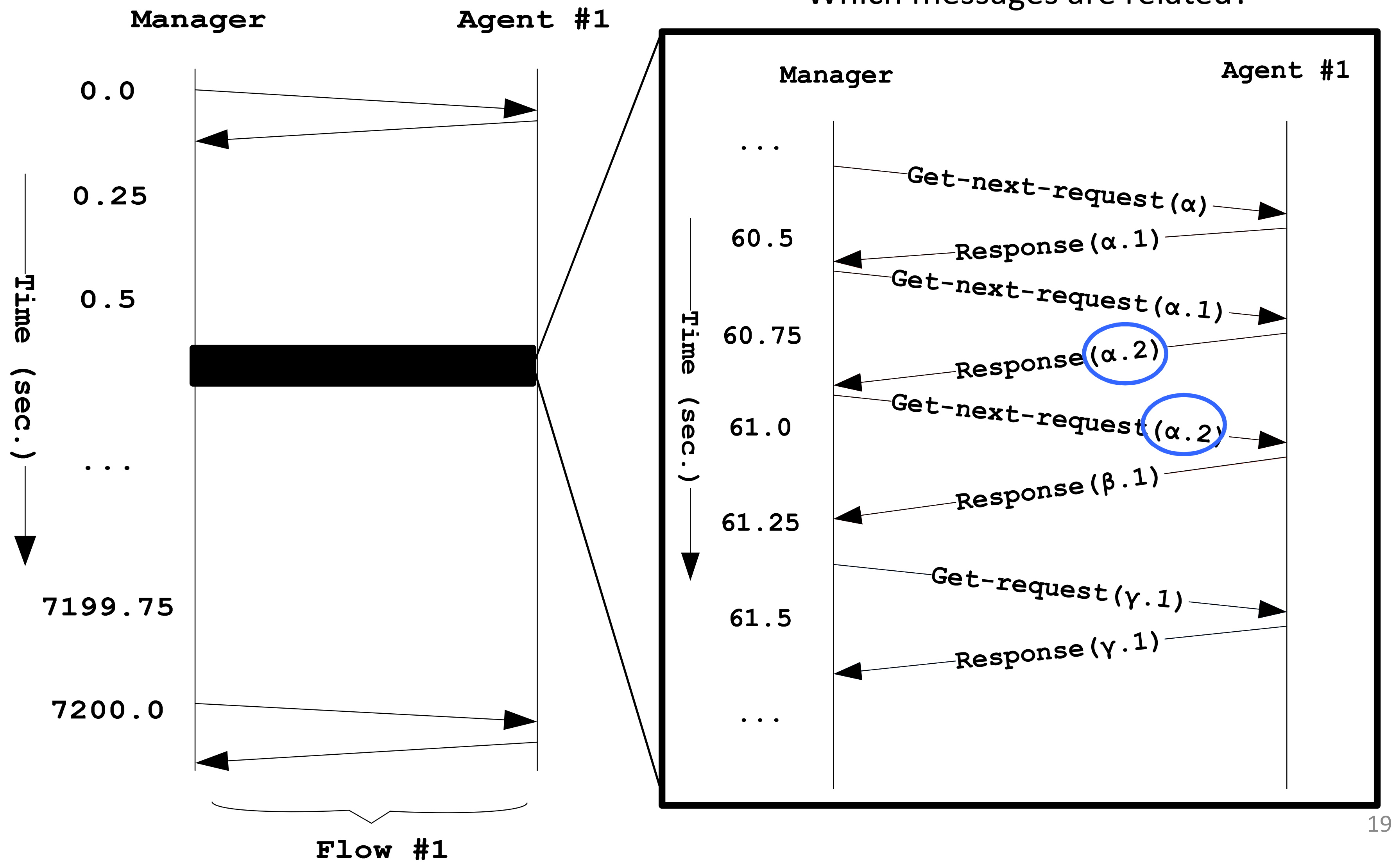
# Part of a single flow

Which messages are related?



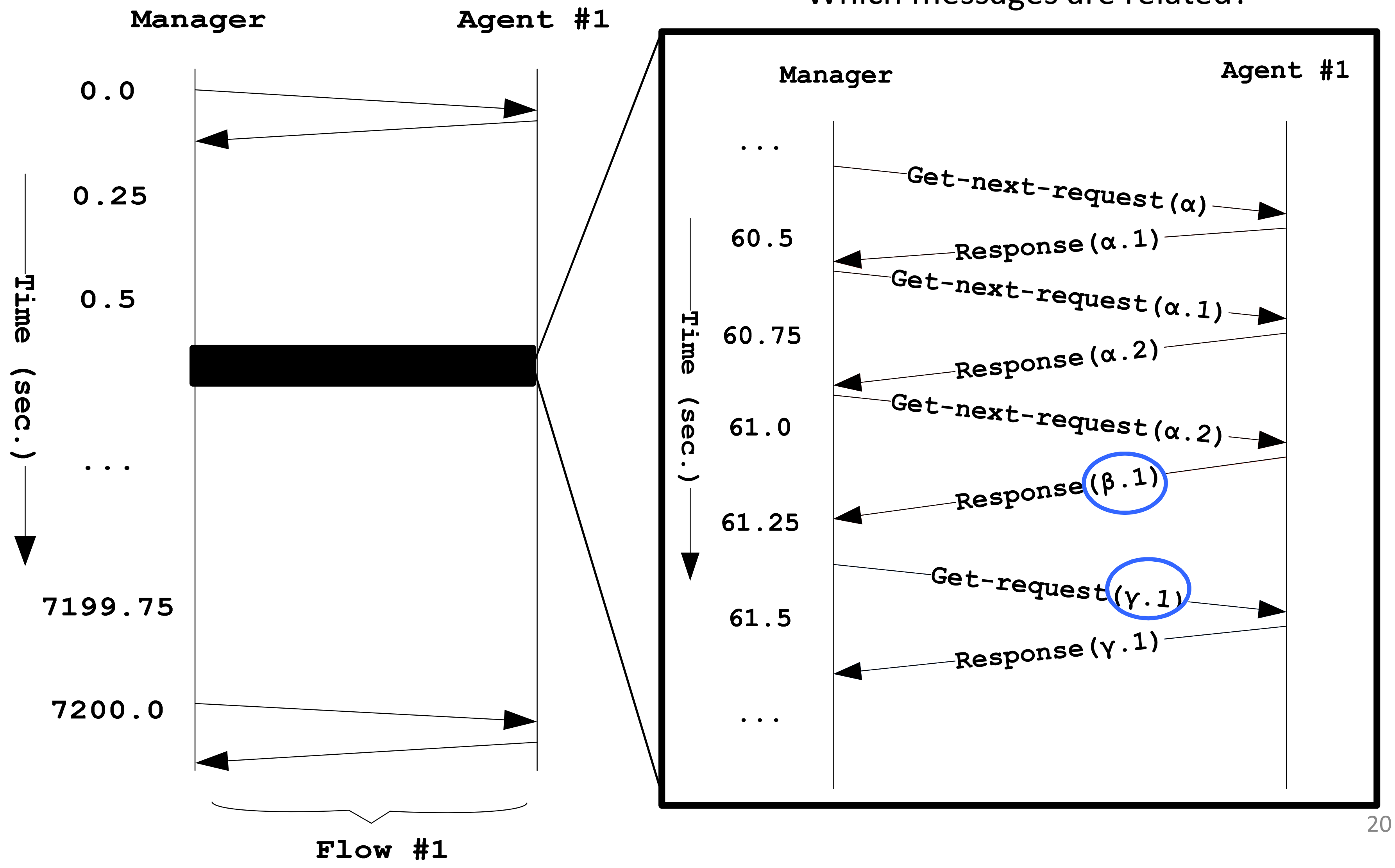
# Part of a single flow

Which messages are related?

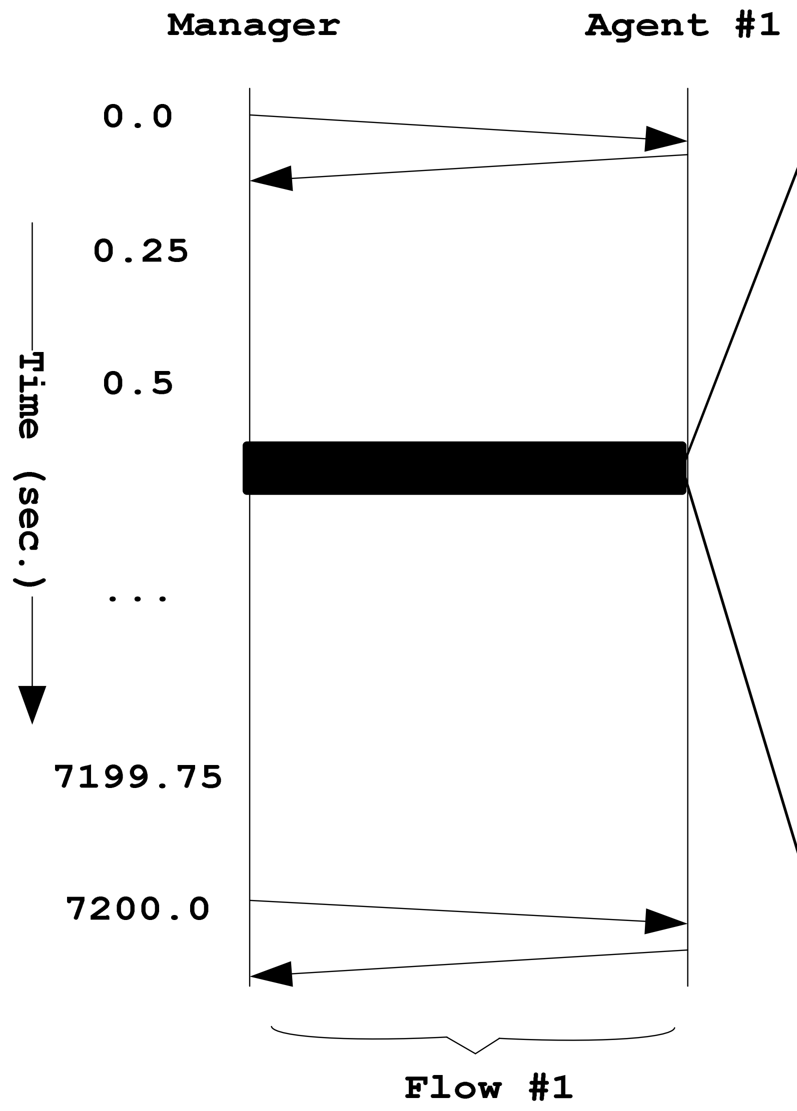


# Part of a single flow

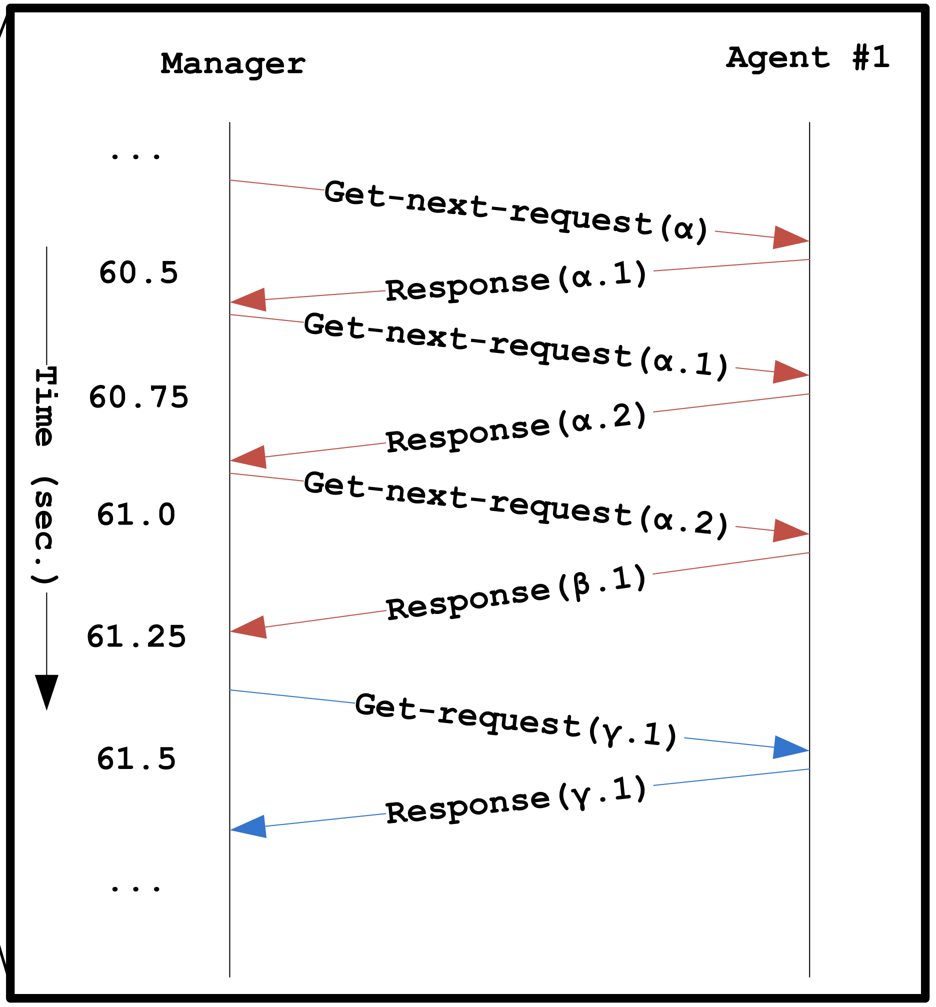
Which messages are related?



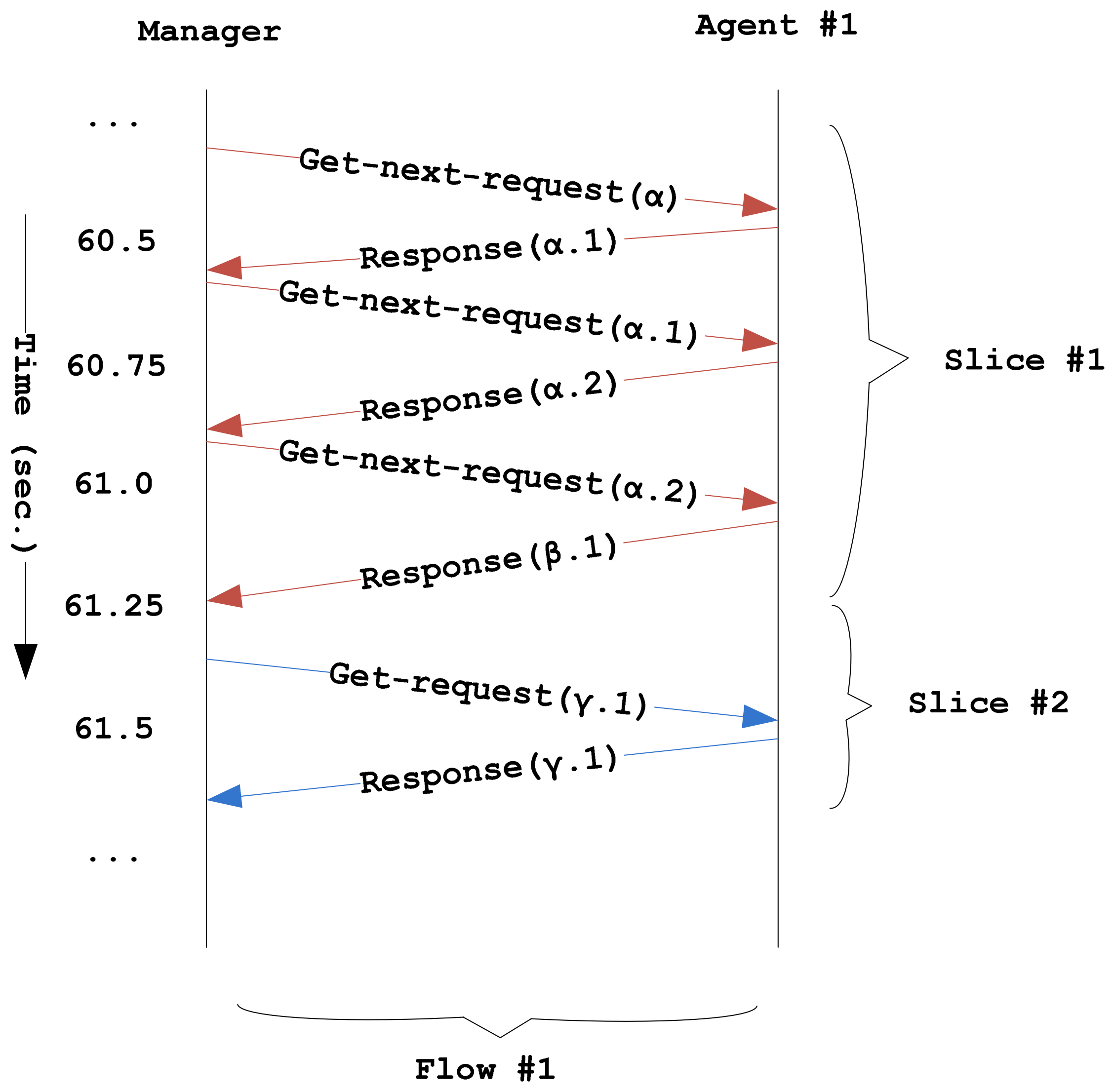
# Part of a single flow



Which messages are related?



## Slices



Manager
Agent #1
...
60.5
60.75
61.0
61.25
61.5
...
Time (sec.)
Get-next-request(α)
Response(α.1)
Get-next-request(α.1)
Response(α.2)
Get-next-request(α.2)
Response(β.1)
Get-request(γ.1)
Response(γ.1)
Slice #1
Slice #2
Flow #1

## Slice (1/3)

- Definition (**slice**): A slice S with parameter e is a subset of SNMP messages in a flow F for which the following properties hold:

  (S1) All messages are exchanged between the same two transport endpoints (a single transport endpoint pair).

  (S2) All non-response messages must have a PDU of the same type.

  (S3) All messages with a PDU of type Get, Set, Trap, or Inform must contain the same set of OIDs.

# Slice (2/3)

- Definition (**slice**): A slice S with parameter e is a subset of SNMP messages in a flow F for which the following properties hold:

  ...

  (S4) Each GetNext or GetBulk message must either contain the same set of OIDs as the preceding request or they must be linked to a response of the last previously answered request, that is the request must contain at least one OID that has been contained in the (repeater) varbind list of a preceding response message of the last answered request message.

# Slice (3/3)

- Definition (**slice**): A slice S with parameter e is a subset of SNMP messages in a flow F for which the following properties hold:

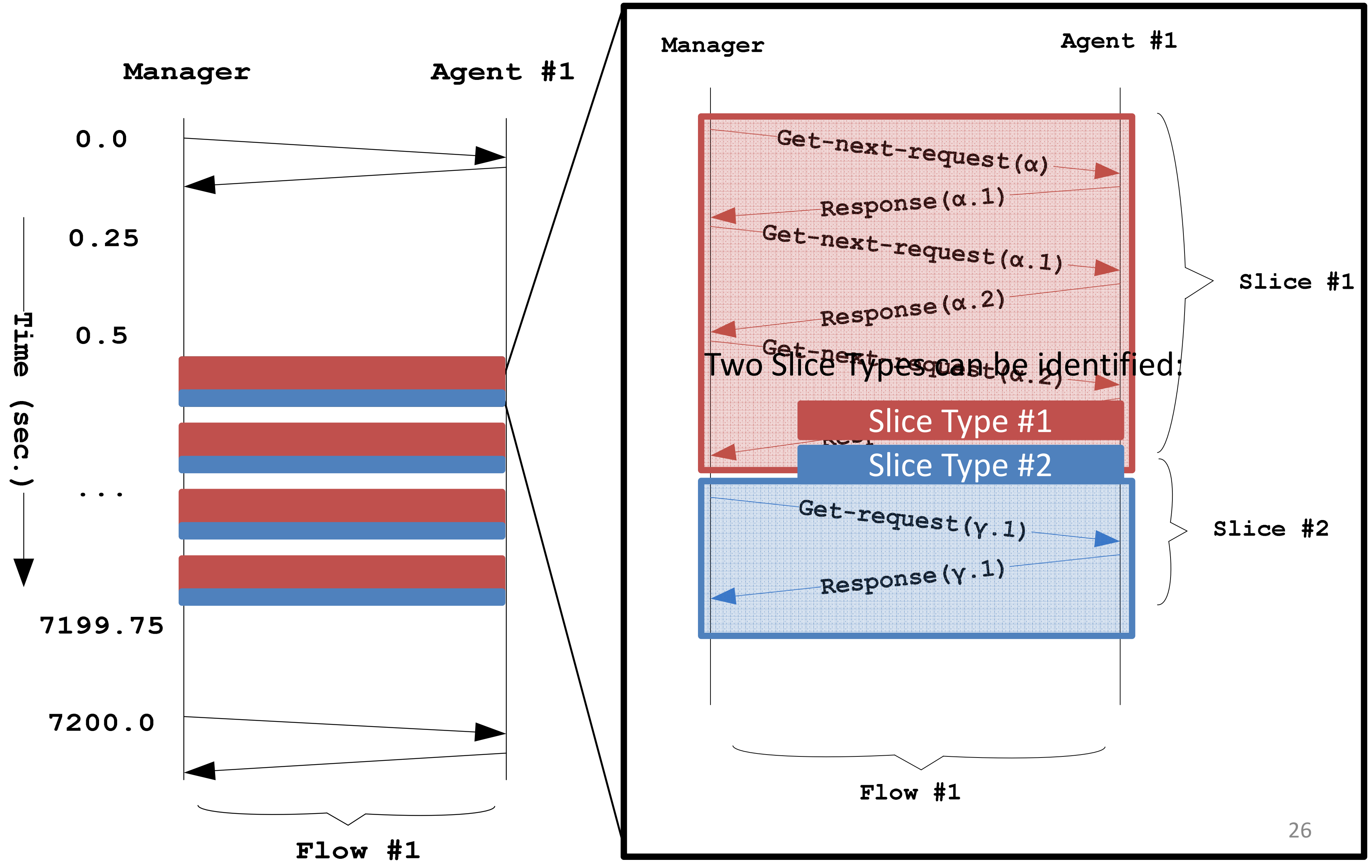
  ...

  (S5) All Response messages must follow a previous request message.

  (S6) For any two subsequent request messages Q1 and Q2 with Q1.time < Q2.time, the following condition must hold:

  $$(Q2.time - Q1.time) < e$$

# Any more relations?



Two Slice Types can be identified:

Slice Type #1

Slice Type #2

## Slice prefix

- Goal: to be able to compare various slices using the slice prefix
- Specific configuration of slice initiator is only relevant -> consider only non-response messages.

# Slice prefix

```
-----------------------------------------------------------------
    Message | Direction | PDU type        | OIDs
-----------------------------------------------------------------
       0        A -> B    GetNext Request   alpha, beta
       1        B -> A    Response          alpha.0, beta.0
       2        A -> B    GetNext Request   alpha.0, beta.0
       3        B -> A    Response          alpha.1, beta.1
       4        A -> B    GetNext Request   alpha.1, beta.1
       5        B -> A    Response          gamma.0, delta.0
-----------------------------------------------------------------
```

# Slice Signature

- Definition (**slice signature**): A slice signature S.sig of a slice S is a set of OIDs derived from the OIDs contained in the non-response messages of a slice. This set S.sig consists of the following OIDs:
- case S.type = GetNext or S.type = GetBulk:

$$S.sig = \bigcup_{\substack{N \text{ in } S \\ N \text{ is non-response}}} (N.oids) - \bigcup_{\substack{R \text{ in } S \\ R \text{ is response}}} (R.oids)$$

- otherwise:

$$S.sig = \bigcup_{\substack{N \text{ in } S \\ N \text{ is non-response}}} (N.oids)$$

# Slice Signature

```
-----------------------------------------------------------------
    Message | Direction | PDU type        | OIDs
-----------------------------------------------------------------
       0         A -> B    GetNext Request   alpha, beta
       1         B -> A    Response          alpha.0, beta.0
       2         A -> B    GetNext Request   alpha.0, beta.0
       3         B -> A    Response          alpha.1, beta.1
       4         A -> B    GetNext Request   alpha.1, beta.1
       5         B -> A    Response          gamma.0, delta.0
-----------------------------------------------------------------
```

S.sig = { alpha, beta, alpha.0, beta.0, alpha.1, beta.1 } **-**

{ alpha.0, beta.0, alpha.1, beta.1, gamma.0, delta.0 } **=**

**{ alpha, beta }**

# Slice Prefix

- Definition (**OID prefix**): An OID a = a_1.a_2...a_n is a prefix of OID b = b_1.b_2...b_m if and only if

  (P1) $n < m$ and

  (P2) $a_i = b_i$ for $1 <= i <= n$.

- Definition (**slice prefix**): The slice prefix S.slice is the set of all OIDs o in S.sig for which there is no p in S.sig such that p is a prefix of o.

```
----------------------------------------------------------------
  Message | Direction | PDU type         | OIDs
------------------------------------------------------------------
     0         A -> B    GetNext Request   alpha, beta,
                                           sysUpTime
     1         B -> A    Response          alpha.1, beta.1,
                                           sysUpTime.0
     2         A -> B    GetNext Request   alpha.1, beta.1
     3         B -> A    Response          alpha.2, beta.3
     4         A -> B    GetNext Request   alpha.2, beta.2
     5         B -> A    Response          alpha.3, beta.3
     6         A -> B    GetNext Request   alpha.3, beta.3
     7         B -> A    Response          alpha.4, gamma.1
     8         A -> B    GetNext Response  alpha.4
     9         B -> A    Response          delta.1
------------------------------------------------------------------
```

S.Sig = { alpha, beta, beta.2, sysUpTime }

S.Prefix = { alpha, beta, sysUpTime }

## Slice Type (1/2)

- Definition (**slice equivalence**): Two slices A and B satisfy the binary slice equivalence relation A ~ B if the following properties hold:

  (EQ1) All messages in A and B have been exchanged between the same two network layer endpoints.

  (EQ2) All read request messages, write request messages, and notification messages in A and B originate from the same network layer endpoint.

  (EQ3) All non-response messages in A and B are of the same type.

  (EQ4) The slices A and B have the same prefix, that is A.prefix = B.prefix.

## Slice Type (2/2)

- It can be easily seen that the relation ~ is reflexive, symmetric, and transitive and thus forms an equivalence relation between slices.

- Definition (**slice type**): Let S be a set of slices, then all slices in the equivalence class [A] = {s in S | s ~ A} with A in S, are of the same slice type.

# Walks (1/2)

- Definition (**walk**): A walk W is a slice S with the following properties:

  (W1) The type of the slice S is either GetNext request or GetBulk request.

  (W2) At least one object identifier in the sequence of requests at the same varbind index must be increasing lexicographically while all object identifiers at the same varbind index have to be non-decreasing.

## Walks (2/2)

- Definition (**strict walk**): A walk W is a strict walk if all object identifiers in the sequence of requests at the same varbind index are strictly increasing lexicographically. Furthermore, the object identifiers at the same index of a response and a subsequent request must be identical.

- Definition (**prefix constrained walk**): A walk W is a prefix constrained walk if all object identifiers at the same index in the request have the same object identifier prefix. This prefix is established by the first request message within the walk.

# Walks

| Message | Direction | PDU type | OIDs |
|---|---|---|---|
| 0 | A -> B | GetNext Request | alpha, beta |
| 1 | B -> A | Response | alpha.0, beta.0 |
| 2 | A -> B | GetNext Request | alpha.0, beta.0 |
| 3 | B -> A | Response | alpha.1, beta.1 |
| 4 | A -> B | GetNext Request | alpha.1, beta.1 |
| 5 | B -> A | Response | gamma.0, delta.0 |

## Analysis results

- Trace #1
  - Number of SNMP messages in trace: 361000
  - Slices found: 6940
    - Get-next-request: 5157
    - Get-request: 1783
  - Slice Types found: 1277
    - Get-next-request: 57
    - Get-request: 1220

## Analysis results

- Trace #2
  - Number of SNMP messages in trace: 15099
  - Slices found: 7504
    - Get-next-request: 192
    - Get-bulk-request: 2785
    - Get-request: 4527
  - Slice Types found: 157
    - Get-next-request: 4
    - Get-bulk-request: 58
    - Get-request: 95

- Questions?

- Next steps:
  - Updating draft based on the received feedback