Learning CoAP separate responses by example
draft-castellani-lwig-coap-separate-response-00

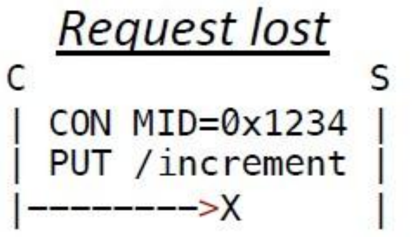
Angelo P. Castellani

Presenter: Xuan He

Purpose

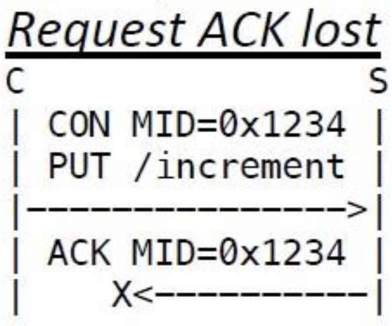
- This draft aims at providing interesting examples of CoAP separate responses that are useful to aid CoAP implementers on understanding possible rare situation incurring.

Taxonomy of cases



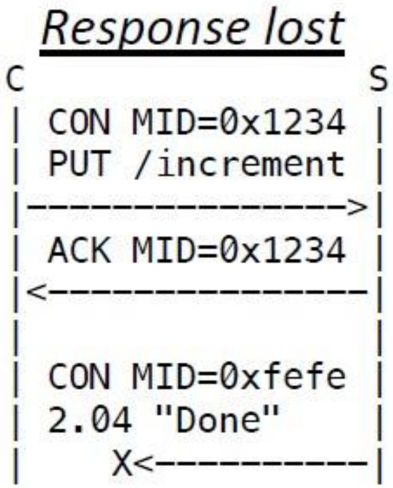
Request lost

```
C                S
| CON MID=0x1234 |
| PUT /increment |
|-------->X      |
```



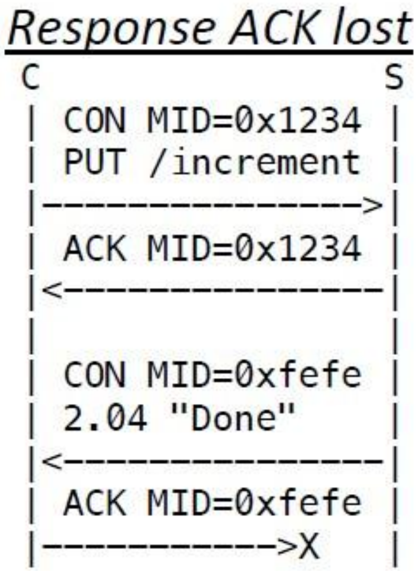
Request ACK lost

```
C                S
| CON MID=0x1234 |
| PUT /increment |
|--------------->|
| ACK MID=0x1234 |
|     X<---------|
```



Response lost

```
C                S
| CON MID=0x1234 |
| PUT /increment |
|--------------->|
| ACK MID=0x1234 |
|<---------------|
|                |
| CON MID=0xfefe |
| 2.04 "Done"    |
|     X<---------|
```

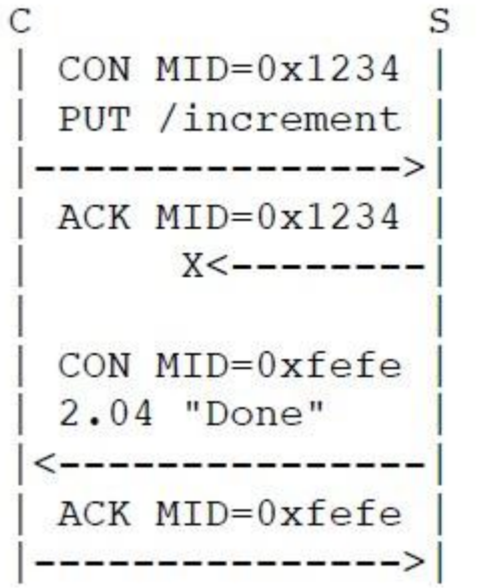


Response ACK lost

```
C                S
| CON MID=0x1234 |
| PUT /increment |
|--------------->|
| ACK MID=0x1234 |
|<---------------|
|                |
| CON MID=0xfefe |
| 2.04 "Done"    |
|<---------------|
| ACK MID=0xfefe |
|---------->X    |
```

In this pictures retransmissions are NOT shown. Still the situations that might occur can be synthetized using this reference cases.

Optimization on request ACK lost

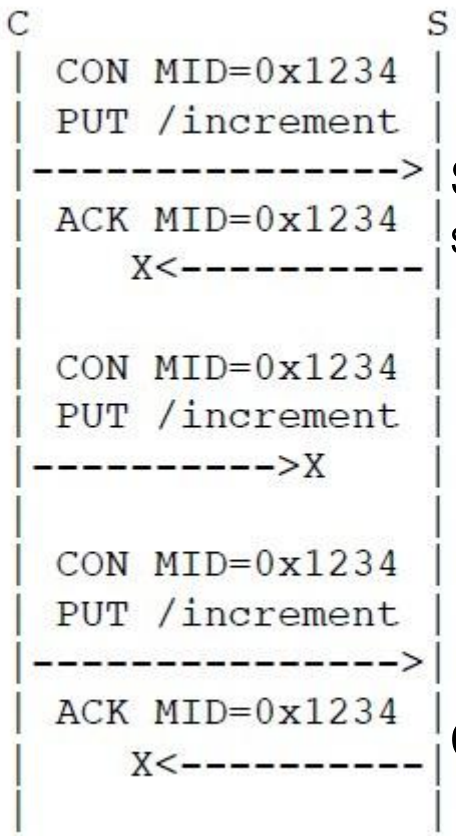


```
C                 S
| CON MID=0x1234  |
| PUT /increment  |
|---------------->|
| ACK MID=0x1234  |
|      X<---------|
|                 |
| CON MID=0xfefe  |
| 2.04 "Done"     |
|<----------------|
| ACK MID=0xfefe  |
|---------------->|
```

Client implementations supporting only the empty Token (no Token support) are encouraged to randomly select local UDP source port at each new request; this implementation shrewdness smoothly resolves confusion.

Always having the Token Option set to a random value realistically resolves any possible confusion in this case, at the obvious cost of its added complexity in the client implementation and network overhead.

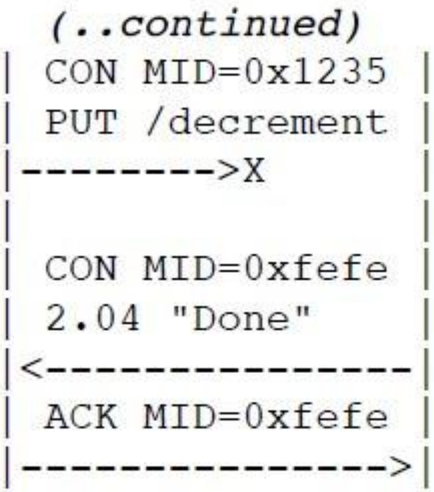
Naïve client



```
C                   S
| CON MID=0x1234 |
| PUT /increment |
|--------------->|
| ACK MID=0x1234 |
|     X<---------|
|                |
| CON MID=0x1234 |
| PUT /increment |
|---------->X    |
|                |
| CON MID=0x1234 |
| PUT /increment |
|--------------->|
| ACK MID=0x1234 |
|     X<---------|
|                |
```

Server processing starts

Client gives up



```
  (..continued)
| CON MID=0x1235 |
| PUT /decrement |
|-------->X      |
|                |
| CON MID=0xfefe |
| 2.04 "Done"    |
|<---------------|
| ACK MID=0xfefe |
|--------------->|
```

Client issues a new request

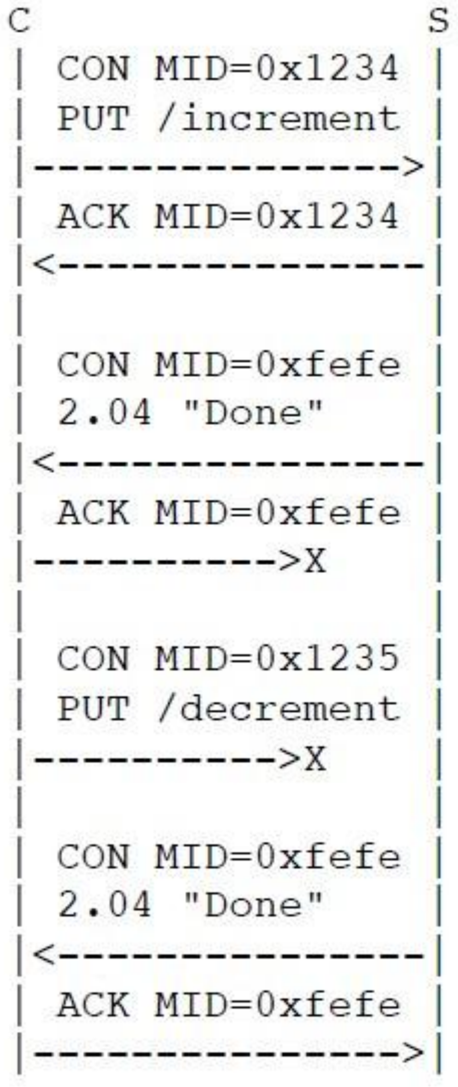
Server processing ends

inconsistency!!

A naive client implementation using the empty Token and a static local UDP port：

This leads to the indication that a client should in general avoid reusing the same session , i.e., [loc-host, loc-port, rem-host, rem-port, token], even if it has failed.

Inexperienced client



```
C                S
| CON MID=0x1234 |
| PUT /increment |
|--------------->|
| ACK MID=0x1234 |
|<---------------|
|                |
| CON MID=0xfefe |
| 2.04 "Done"    |
|<---------------|
| ACK MID=0xfefe |
|---------->X    |
|                |
| CON MID=0x1235 |
| PUT /decrement |
|---------->X    |
|                |
| CON MID=0xfefe |
| 2.04 "Done"    |
|<---------------|
| ACK MID=0xfefe |
|--------------->|
```

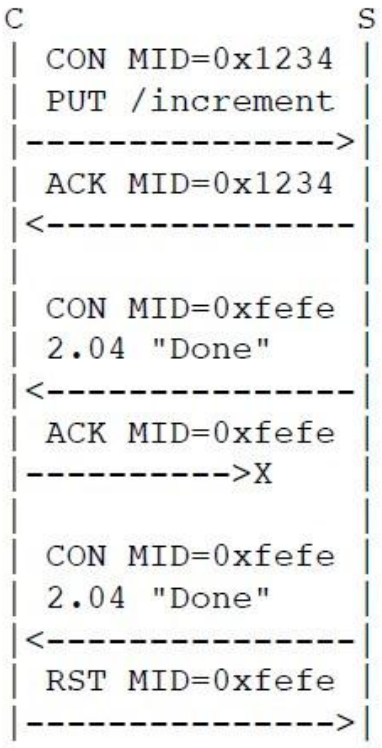
client issues a new request

server retransmits the response

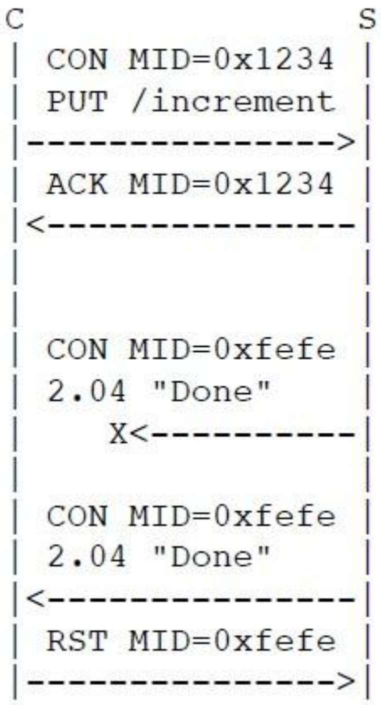
client deduplication did not work!

An inexperienced client not having robust deduplication in place and reusing the same session.

Forgetful vs Rebooting client



```
C                 S
| CON MID=0x1234 |
| PUT /increment |
|--------------->|
| ACK MID=0x1234 |
|<---------------|
|                |
| CON MID=0xfefe |
| 2.04 "Done"    |
|<---------------|
| ACK MID=0xfefe |
|---------->X    |
|                |
| CON MID=0xfefe |
| 2.04 "Done"    |
|<---------------|
| RST MID=0xfefe |
|--------------->|
```



```
C                 S
| CON MID=0x1234 |
| PUT /increment |
|--------------->|
| ACK MID=0x1234 |
|<---------------|
|                |
|                |
| CON MID=0xfefe |
| 2.04 "Done"    |
|    X<----------|
|                |
| CON MID=0xfefe |
| 2.04 "Done"    |
|<---------------|
| RST MID=0xfefe |
|--------------->|
```

client goes down for reboot

client is up again

An optimistic server implementation might think that the client has received the response even if it has replied with a RST.

Open issue:
Should the server change its behavior depending on the fact that it received a RST instead of an ACK?

Next Step

- Comments and Questions?

- WG Adoption?