OCP: OPES callout protocol

`draft-ietf-opes-ocp-core-00.txt`

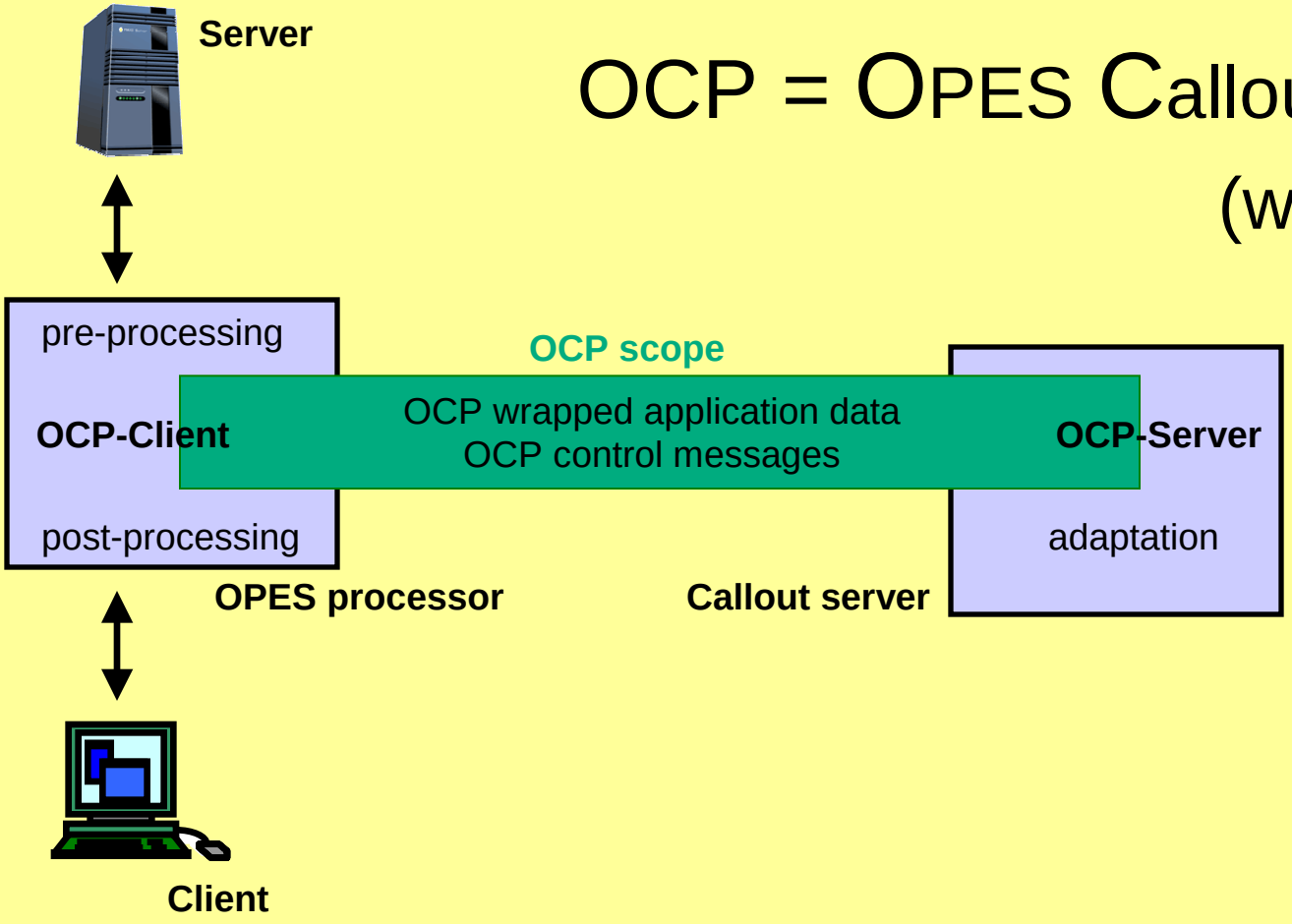
OPES WG meeting on 57th IETF in
Wien, Österreich

Martin Stecher (martin.stecher@webwasher.com)
Alex Rousskov (rousskov@measurement-factory.com)

Content

- OCP overview
  - What is OCP?
  - OCP scope, principles, transport
- Progress since last meeting
- OCP details
  - Message format / framing
  - Asynchronous, Concurrent, Data preservation
  - Negotiation
- Current and future
  - Latest status, implementation efforts
  - Next steps

What is OCP?



OCP = OPES Callout Protocol

(working title)

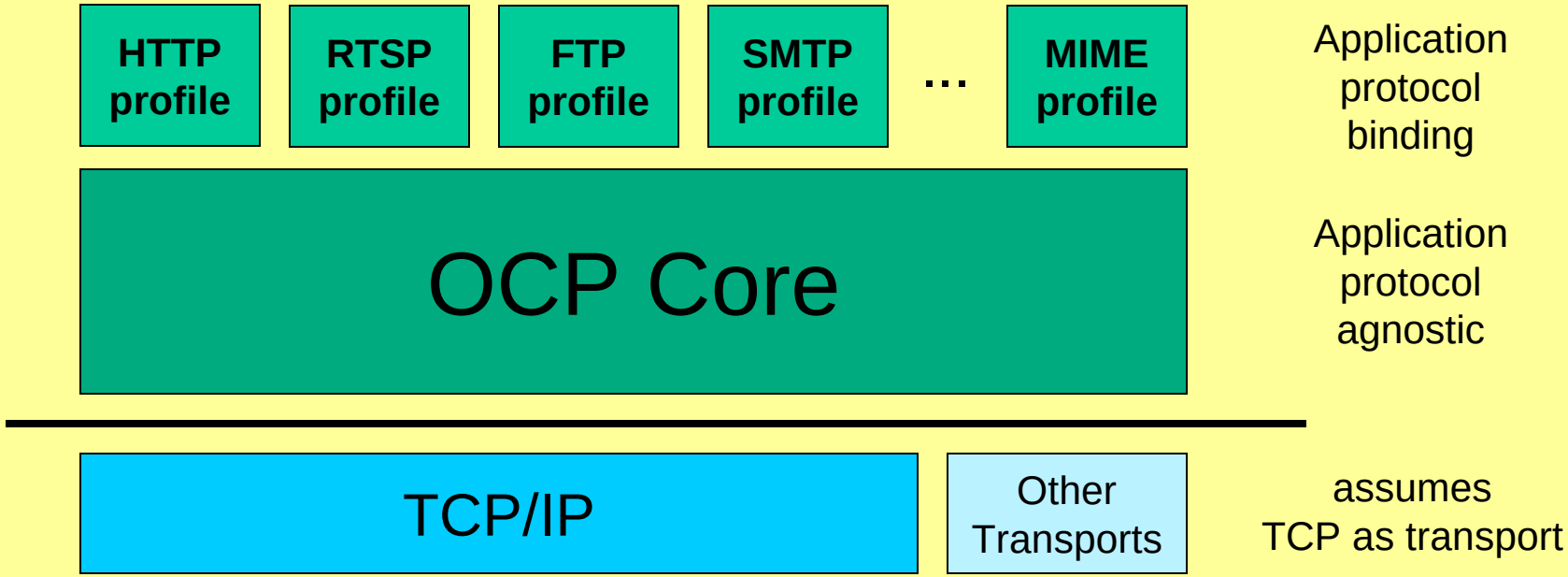
OCP Scope

- Application message exchange between OCP client and server? ☑ **yes**
- Application message pre- and post-processing? ☒ **no**
- Iteration over OPES services? ☒ **no**
- Specific adaptation features? ☒ **no**
- Message exchange optimization? ☑ **yes**
- Message adaptation optimization? ☒ **no**

OCP design principles

- Application-agnostic Core
- Supports exchange of application messages
- Small overhead
- Non-blocking
- Concurrency
- Minimal RISC-style set of features facilitating adaptations

OCP Building Blocks



OCP Transport

- Require reliable ordered delivery of messages
- Assume TCP/IP
- Do not discourage other protocols
- This “lazy” approach adopted from HTTP RFC 2616

Progress since last meeting (1)

- Clarified scope
- Decided on OCP transport
  - Long disussion on the mailing list in April/May.
  - Group looked at BEEP, SOAP, HTTP, OCPTRAN, TCP, multiple transport bindings
  - Consensus: Moving forward with a custom-tailored transport for OCP, then evaluate that approach
  - OCPTRAN is not defined as a standalone thing but worked into OCP Core (on top of TCP).
- Mostly decided on OCP message format/framing

Progress since last meeting (2)

- Added negotiation and capability inquiry mechanisms
- Merged application data and metadata
- Simplified protocol by tying OCP and transport connections
- Started “HTTP adaptation with OPES” draft
- Started client and server implementation for HTTP application!

Message format / framing

- Strict, simple grammar but MIME-looking
- Integral types: integers and NetString-like strings
- Complex types: {structures} and (lists) Limit nesting(?)
- Strive for consistent, clean design

ABNF grammar and examples (1)

Atoms

```
atom = bare-value / quoted-value
bare-value = 1*safe-OCTET                ; used as number or string
safe-OCTET = ALPHA / DIGIT / "-" / "_"
quoted-value = DQUOTE data DQUOTE
data = size ":" <n>OCTET                 ; <n> == size
```

Examples:

```
5
2147483647
Hello
"5:Hello"
"19:http://www.abc.org/"
"0:"
```

ABNF grammar and examples (2)

Structures and Lists

```
structure = "{" [ value *(SP  value) ] "}"   ; spaced values
list =      "(" [ value *("," value) ] ")"   ; comma-separated values
```

Examples:

```
{x 5}
("19:http://www.abc.org/","19:http://www.xyz.net/")
{one}
()
{(23,5) {nested}}
```

Values

```
value = atom / structure / list
```

ABNF grammar and examples (3)

Messages

```
message = name [anonym-parameters]
            [named-parameters]
            [payload]
            ";" CRLF

  name = ALPHA *safe-OCTET
  anonym-parameters = 1*(SP   anonym-parameter) ; spaced parameters
  named-parameters =  1*(CRLF named-parameter)  ; CRLF-separated params
  anonym-parameter = value
  named-parameter = name ":" SP value
  payload = CRLF data
```

ABNF grammar and examples (4)

Message Examples

- `CS;`
- `SGC 10 ({"22:myserver.com/my-filter"});`
- `TS 88 10;`
-
```
DUM 88 1 0
Kept: 0
AM-Part: response-body
26:<html>
<body>
This is my;
```

A typical message flow

Client	Server	
CS ...	CS ...	**start connection**
NO ...	NR ...	**profile negotiation**
SGC ...		**create service group**
TS ...	TS ...	**start transaction**
AMS ...	AMS ...	**application message start**
DUM ...		**first app. message fragment**
DUM ...	DUM ...	**next fragment, first answered**
DUM ...	DUY ...	**last fragment, use data unchanged**
AME ...	AME ...	**end of application message**
TE ...	TE ...	**transaction end**
TS ...	TS ...	**next transaction starts**

Asynchronous data flow

- An agent can send any number of DUM messages (other messages as well) without waiting for a reply
- The other agent can request a data pause if needed

Concurrency

Agents can multiplex transactions on one connection

```
CS;
TS 1 ...;
AMS 1 1;
DUM 1 1 0 ... ;
TS 2 ...;
AMS 2 1;
DUM 1 1 100 ... ;
DUM 2 1 100 ... ;
TS 3 ...;
DUM 2 1 200 ... ;
AMS 3 1;
DUM 2 1 300 ... ;
DUM 3 1 0 ... ;
DUM 1 1 200 ... ;
```

Data Preservation (1)

- OPES processor can keep a copy of the data sent to the callout server and allow the server to refer to it.
- Do not mix data preservation with caching:
  - Preservation is a firm commitment
  - Caching is an opportunistic optimization

Data Preservation (2)

- Complete fragments are kept, any byte range can be referred.
- Named parameters “Kept” and “Wont-Use” in DUM and DUY messages.
- Data preservation can start and stop with every fragment.
- Default: Free preserved data after reference, can be changed: Don’t free or free more data

Data Preservation Example (1)

```
DUM 88 1 0
5:Mary ;
DUM 88 1 5
Kept: 0
3:had;
DUM 88 1 8
Kept: 3
9: a little;
DUM 88 1 17
5: lamb;
```

Application message sent so far:

Mary **had a little** lamb (Bold letters are preserved)

Data Preservation Example (2)

One way to replay the message

`Mary` **`had a little`** `lamb` (Bold letters are preserved)

```
DUM 88 1 0
5:Mary ;
DUY 88 1 0 12;
DUM 88 1 5
5: lamb;
```

preservation commitment stops

Data Preservation – Wont-Use

- By default, preservation commitment is terminated for kept octets below the last refered octet in a DUY message received.
- Wont-Use parameter in DUM or DUY message can change this.
  - Can refer to same octets again
  - Can terminate commitment for more than the referred octets

Wont-Use examples

```
DUY 1 1 0 200
Wont-Use: 300;
```
commitment terminates for octects 0-300

```
DUM 1 1 0 3
Wont-Use: 4294967295
3:abc;
```
will not use any kept octet

```
DUY 1 1 0 200
Wont-Use: 0;
```
continue to keep the first 200 octets

Negotiation

- Either agent can initiate
- Can happen at any time
- Must happen at the beginning to negotiate encryption and application specifics
- Collision resolution (OPES processor has priority)

Negotiation Example

- OPES processor offers two profiles (extensions of OCP Core)

```
NO ({"38:http://iana.org/opes/ocp/HTTP/response"},
{"29:http://iana.org/opes/ocp/MIME"});
```

- Callout server chooses HTTP/OCP for HTTP responses

```
NR {"38:http://iana.org/opes/ocp/HTTP/response"};
```

Negotiation scope (1)

- Negotiations can either be connection-globally or per service group
- Normal NO/NR messages for connection-global negotiation

  ➔ Switch to profile when NR message is sent

Negotiation scope (2)

- If per service group, NO/NR messages have an additional, optional, named parameter; specified “sg-id”

- Transaction-Start (TS) messages have mandantory anonymous sg-id parameter

  ➔ If profile has been negotiated for that sg-id earlier, transaction will use it

Negotiation Example (2)

- Connection globally negotiation

```
CS;
```

connection starts without profile, plain OCP Core

```
NO ({"38:http://iana.org/opes/ocp/HTTP/response"},
{"29:http://iana.org/opes/ocp/MIME"});

NR {"38:http://iana.org/opes/ocp/HTTP/response"};
```

from here on connection will use HTTP-Response profile

Negotiation Example (3)

- Per service group negotiation

```
SGC 10 ({"22:myserver.com/my-filter"});
```

service group containing single service „my-filter“

```
NO ({"38:http://iana.org/opes/ocp/HTTP/response"},
{"29:http://iana.org/opes/ocp/MIME"})
SG: 10;

NR {"38:http://iana.org/opes/ocp/HTTP/response"}
SG: 10;
```

negotiation for service group 10, no profile switch here

```
TS 88 10;
```

transaction starts for service group 10, using profile HTTP-response

Capability inquiry

- Can You Support (can-you)?
  I Support (i-can)!
- Do You Currently Use (do-you)?
  I Currently Use (i-do)!
- Perhaps too much complexity

Implementation Effort

- Start with adapting HTTP
- Not a reference implementation (yet), but a proof of concept
- Keeping performance goals and competition with ICAP in mind

Future

- Polish Core
- Work on “HTTP adaptation with OPES” draft
- Start working on “SMTP adaptation with OPES” draft
- Share and take into account HTTP/OCP implementation lessons
- Add Transport Level Security support