



Application Flow Control with iRules – HTTP

James Quinby– MC

David Larsen– Presenter

Stephen Anderson – Presenter

John Eudy - Presenter

John Alam – Presenter



Intro:



WHY HTTP IRULES?

WHAT'S IN IT FOR ME?

WHO ARE YOU TO TELL ME HOW TO DO IRULES AND IS THIS WORTH MY TIME?



Agenda: Application Flow Control with iRules - HTTP

- **iRules Level Set**
- **HTTP Protocol Review**
- **HTTP Request Side Overview**
- **HTTP Response Side Overview**
- **HTTP Related Events**
- **HTTP Headers**
- **STREAM Command**

iRules: Level Set

- **First rule of iRules – make sure you need an iRule.**
- **Comment code**
- **Use appropriate looping**
- **Use efficient criteria selection methods**
- **Use datagroups**

Protocol



The HyperText Transfer Protocol

- HTTP is the ***stateless***, text-based protocol responsible for the ***World Wide Web***, and has just one purpose: the transfer and delivery of HTTP messages.
- The basic message units are ***requests*** and ***responses***.
- Inside those messages are:

Methods / Status / Addressing / Version

Headers

Payload



The HyperText Transfer Protocol

- Let's start by reviewing a basic HTTP message transfer and delivery.

- A client will make a request:

```
GET /index.html HTTP/1.1
Host: www.example.com
Accept: text/html
Accept-Encoding: x-zip; x-compress
User-Agent: libwww/1.3.1
If-Modified-Since: Thu, 07-Apr-2011 12:00:00 GMT
```

- And a server will respond:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Server: Apache
Date: Thu, 07 Apr 2011 19:10:53 GMT
Content-Encoding: gzip

<html><head><title>...
```



The HyperText Transfer Protocol: request

- To retrieve a document (via request message) from the following resource:

`http://x.y.com/path/file.html?user=123&is=me#sometimes`

The diagram shows the URL `http://x.y.com/path/file.html?user=123&is=me#sometimes` with brackets underneath identifying its parts: `http` is the scheme, `//x.y.com` is the address, `/path/file.html` is the path/resource, `?user=123&is=me` is the query string, and `#sometimes` is the fragment.

- Your HTTP (browser) client will generate a request message:

`GET /path/file.html?user=123&is=me HTTP/1.1`

The diagram shows the request message `GET /path/file.html?user=123&is=me HTTP/1.1` with brackets underneath identifying its parts: `GET` is the method, `/path/file.html?user=123&is=me` is the uri, and `HTTP/1.1` is the version. The uri is further broken down into `/path/file.html` (path) and `?user=123&is=me` (query).



The HyperText Transfer Protocol: request

- This resource request line will be followed by a series of **headers** - directives in name/value pairs that tell the server about the client.

```
GET /path/file.html?user=123&is=me HTTP/1.1
```

```
Host: x.y.com
```

```
User-Agent: libwww/1.3.0
```

```
Accept: text/html
```

```
Accept-Encoding: gzip, deflate
```

```
Keep-Alive: 300
```

```
Cookie: mycookie=12345
```

```
If-Modified-Since: Thu, 07 Apr 2011 12:00:00 GMT
```



The HyperText Transfer Protocol: request

- And if there's any payload data, that will commence after a single empty line.

```
GET /path/file.html?user=123&is=me HTTP/1.1
Host: x.y.com
User-Agent: libwww/1.3.0
Accept: text/html
Accept-Encoding: gzip, deflate
Keep-Alive: 300
Cookie: mycookie=12345
If-Modified-Since: Thu, 07 Apr 2011 12:00:00 GMT
```

Other request methods will include payload data at this point



The HyperText Transfer Protocol: request

- Request for Comments (RFC) 2616 defines 8 HTTP request methods.

GET, POST, PUT, DELETE, TRACE, HEAD, OPTIONS, CONNECT

- A **GET** request is generally accompanied by an empty body because the request URI has enough information to complete the method. Semantically, a GET means to *give me a resource*.
- A **POST** request typically contains a body of URL-encoded name/value pairs and does not usually contain large query strings. Semantically, a POST means to *put a resource*.

POST /formlogin.php HTTP/1.1
<headers>

abc=123&is=me&you=arecool



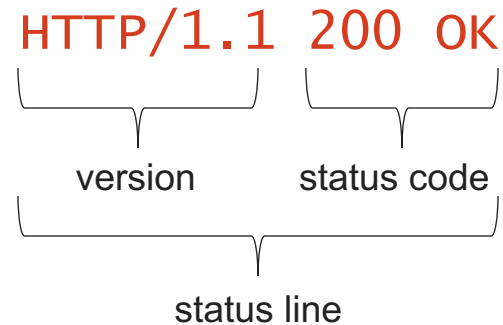
The HyperText Transfer Protocol: response

- The web server then responds with an HTTP response message:

HTTP/1.1 200 OK

version status code

status line

A diagram illustrating the components of an HTTP status line. The text "HTTP/1.1 200 OK" is shown in red. Below it, two curly braces are used for labeling: one under "HTTP/1.1" labeled "version", and another under "200 OK" labeled "status code". A third, larger curly brace spans the entire "HTTP/1.1 200 OK" text and is labeled "status line" below it.

- The status code indicates to the client how the request was processed, and optionally gives an indication for further actions.



The HyperText Transfer Protocol: response

- This response status line will be followed by a series of **headers** - directives in name/value pairs that tell the client about the response.

HTTP/1.1 200 OK

Date: Thu, 07 Apr 2011 19:12:35 GMT

Server: Apache/2.2.9

Last-Modified: Thu, 31 Mar 2011 12:32:00 GMT

Content-Length: 3600

Content-Type: text/html

Content-Encoding: gzip

Set-Cookie: mycookie=12345; path=/;



The HyperText Transfer Protocol: response

- And if there's any payload data, that will commence after a double CRLF.

```
HTTP/1.1 200 OK
Date: Thu, 07 Apr 2011 19:12:35 GMT
Server: Apache/2.2.9
Last-Modified: Thu, 31 Mar 2011 12:32:00 GMT
Content-Length: 3600
Content-Type: text/html
Content-Encoding: gzip
Set-Cookie: mycookie=12345; path=/;
```

```
<html><head><title>Test Page</title></head><body>...
```



The HyperText Transfer Protocol: response

- Per the HTTP RFCs, there are a fair number of response codes. The following are a small few:

100 Continue

200 OK

301 Moved Permanently

302 Moved Temporarily

304 Not Modified

404 Not Found

500 Internal Server Error



and



414

Request-URI Too Long

Events

Traffic Flow Occurrences

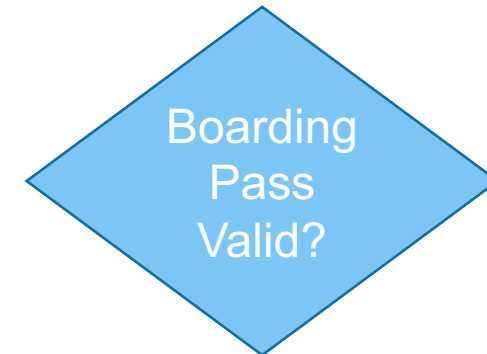
Direct and Manage traffic chronologically

Dynamic decision making based on the state of the Flow.

Where else might we
encounter traffic FLOWS?

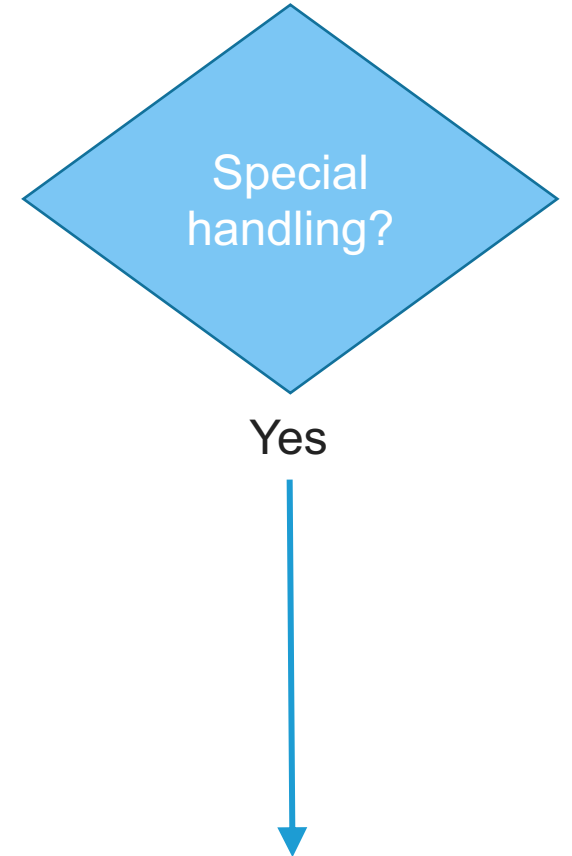
Does the traffic follow
sequential steps?

Let's look at something
we know very well.



Yes





FLIGHT INFORMATION

✈ Departures						
DESTINATION	AIRLINE	FLIGHT	TIME	GATE	STATUS	
ALTOONA	American	46	2:05 PM	C32	On Time	
ATLANTA	American	3074	3:05 PM	D78	On Time	
ATLANTA	American	3069	3:02 PM	D78	On Time	
BALTIMORE	Southwest	3482	12:30 PM	A3	Boarding	
BRADFORD	American	44	2:15 PM	C32	On Time	
CHARLOTTE	American	9405	1:04 PM		New 2:00P	
CHARLOTTE	American	3745	3:57 PM	B33	On Time	
CHARLOTTE	American	3895	3:30 PM	B34	On Time	
CHICAGO/MIDWAY	Southwest	389	2:40 PM	A3	On Time	
CHICAGO/O'HARE	Delta	6957	2:08 PM	B37	On Time	
CORPUS CHRISTI	Southwest	588	2:40 PM	A3	On Time	
DALLAS/DFW	Delta	6725	12:40 PM	B38	On Time	
DALLAS/DFW	American	1426	3:18 PM	B38	On Time	
DALLAS/DFW	Southwest	3385	3:05 PM	A1	On Time	
DENVER	American	799	4:13 PM	A2	On Time	
DETROIT	American	3911	3:38 PM	D68	New 2:25P	
FRANKLIN	American	84	2:05 PM	C32	Cancelled	
FT. MYERS	Southwest	1488	3:30 PM	A3	On Time	
HAGERSTOWN	American	52	2:25 PM	C32	On Time	
HOUSTON/HOBY	Southwest	389	2:40 PM	A3	On Time	
HOUSTON/IAH	American	3653	11:12 AM	A6	New 1:50P	
HOUSTON/IAH	American	6966	2:05 PM	A6	On Time	
JAMESTOWN	American	34	1:58 PM	C32	On Time	
						12:36 PM

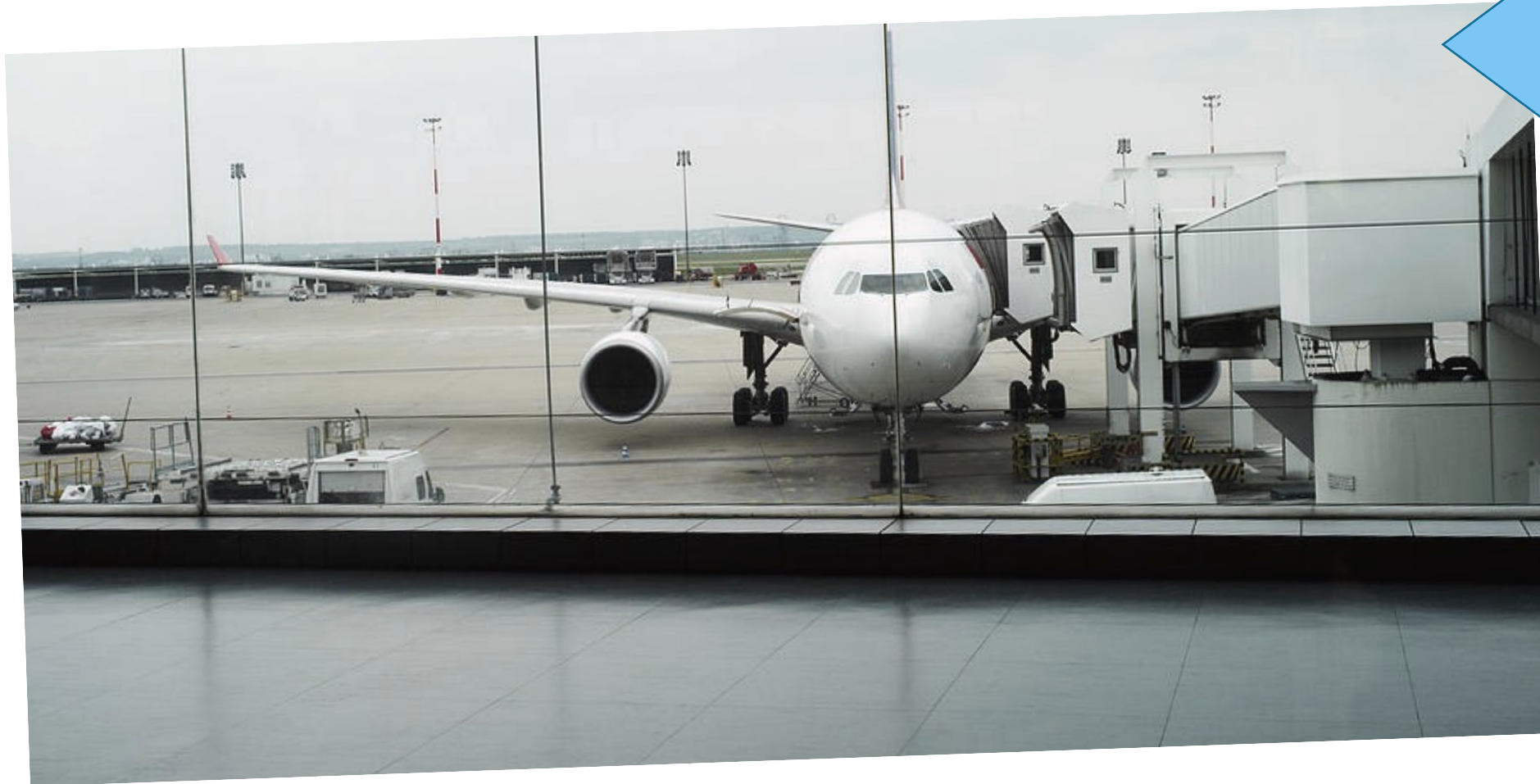
✈ Departures						
DESTINATION	AIRLINE	FLIGHT	TIME	GATE	STATUS	
LANCASTER	American	32	12:30 PM	C32	On Time	
MEMPHIS	Southwest	3482	11:50 PM	A3	Boarding	
MILWAUKEE	Delta	3384	4:00 PM	D63	On Time	
NEW YORK/LKE	Delta	6161	12:34 PM	B40	Boarding	
NEW YORK/LKE	American	3084	12:40 PM	D62	New 1:15P	
NEW YORK/LKE	American	7123	3:27 PM	D62	On Time	
NEW YORK/LGA	American	4981	12:34 PM	B27	Delayed	
NEW YORK/LGA	American	4010	3:04 PM	B28	On Time	
NEW YORK/LGA	American	7223	3:40 PM	D76	Cancelled	
NEW YORK/LGA	American	4851	3:28 PM	D69	On Time	
NEW YORK/LGA	American	3872	3:38 PM	B29	On Time	
NEWARK	American	4224	3:48 PM	A8	On Time	
NEWARK	American	3430	3:10 PM	A8	On Time	
OAKLAND	Southwest	2027	3:38 PM	A1	On Time	
OKLAHOMA CITY	Southwest	389	2:40 PM	A3	On Time	
ONTARIO	Southwest	2071	3:38 PM	A1	On Time	
PHILADELPHIA	American	3900	1:10 PM	B27	On Time	
PHILADELPHIA	American	4842	3:27 PM	B26	On Time	
PHOENIX	Southwest	3877	3:38 PM	A1	On Time	
RALPH-DURHAM	American	4578	3:08 PM	B39	On Time	
TOLSON CITY	American	354	4:30 PM	C38	On Time	
WASHINGTON/DCA	American	52	2:25 PM	C32	On Time	
WASHINGTON/DCA	American	4628	2:40 PM	A4	On Time	
						12:36 PM

✈ Arrivals						
ORIGIN	AIRLINE	FLIGHT	TIME	GATE	STATUS	
ALTOONA	American	75	11:05 AM	C32	On Time	M
ATLANTA	Delta	1599	12:05 PM	D78	In Route	L
ATLANTA	Delta	2100	2:02 PM	D78	New 2:15P	L
BALTIMORE	Southwest	486	4:00 PM	A1	On Time	L
BOSTON	American	4028	12:32 PM	B26	On Time	A
BRADFORD	American	69	3:05 PM	C32	On Time	M
CHARLOTTE	American	3825	3:18 PM	B32	On Time	B
CHARLOTTE	American	3842	3:45 PM	B34	On Time	B
CHICAGO/MIDWAY	Southwest	3452	12:25 PM	A3	Arrived	F
CHICAGO/MIDWAY	American			A3	On Time	F
CHICAGO/O'HARE	American			A8	On Time	F
CHICAGO/O'HARE	American			B27	On Time	C
CHICAGO	American			B28	New 2:50P	F
COLUMBUS	American			A1	On Time	B
DALLAS/DFW	American			B38	On Time	C
DALLAS/DFW	American			B38	On Time	C
DENVER	American			A2	New 2:40P	F
DENVER	American			A3	On Time	B
DETROIT	American					F
JAMESTOWN	American	75	3:10 PM	C32	On Time	M
MEMPHIS	American	4204	12:04 PM	B40	On Time	C
MIDLAND	Southwest	684	2:30 PM	A1	On Time	B
MINNEAPOLIS	American	6399	1:05 PM	D64	In Route	L
NEW ORLEANS	American	9877	2:55 PM		On Time	A
NEW YORK/LKE	American	2914	3:38 PM	D62	New 2:15P	L
NEW YORK/LGA	American	4762	12:01 PM	D62	In Route	L
NEW YORK/LGA	American	4842	12:30 PM	B28	Delayed	L
NEW YORK/LGA	American	3873	3:25 PM	B29	On Time	B
NEW YORK/LGA	American	7022	2:10 PM	D69	New 2:40P	L
NEWARK	American	3987	2:40 PM	A8	New 2:50P	F
OKLAHOMA CITY	American	6134	2:08 PM	A6	New 2:10P	F
OKLAHOMA CITY	Southwest	2077	12:01 PM	A1	In Range	B
PHILADELPHIA	American	3900	1:01 PM	B27	On Time	A
PHILADELPHIA	American	4842	3:01 PM	B26	On Time	A
PHOENIX	American	489	3:11 PM	B26	On Time	A
PROVIDENCE	American	4549	1:45 PM	A8	On Time	F
PUNTA CANA	American	3882	3:45 PM	C38	On Time	F
RALPH-DURHAM	American	4525	12:18 PM	B39	On Time	C
TAMPA	Southwest	1482	2:41 PM	A3	On Time	E
TORONTO CITY	American	110	3:50 PM	C38	On Time	J
WASHINGTON/DCA	American	51	12:06 PM	C32	On Time	M
WASHINGTON/DCA	American	4811	2:08 PM	A4	New 1:54P	F
						12:36 PM

Next connection found?

Yes



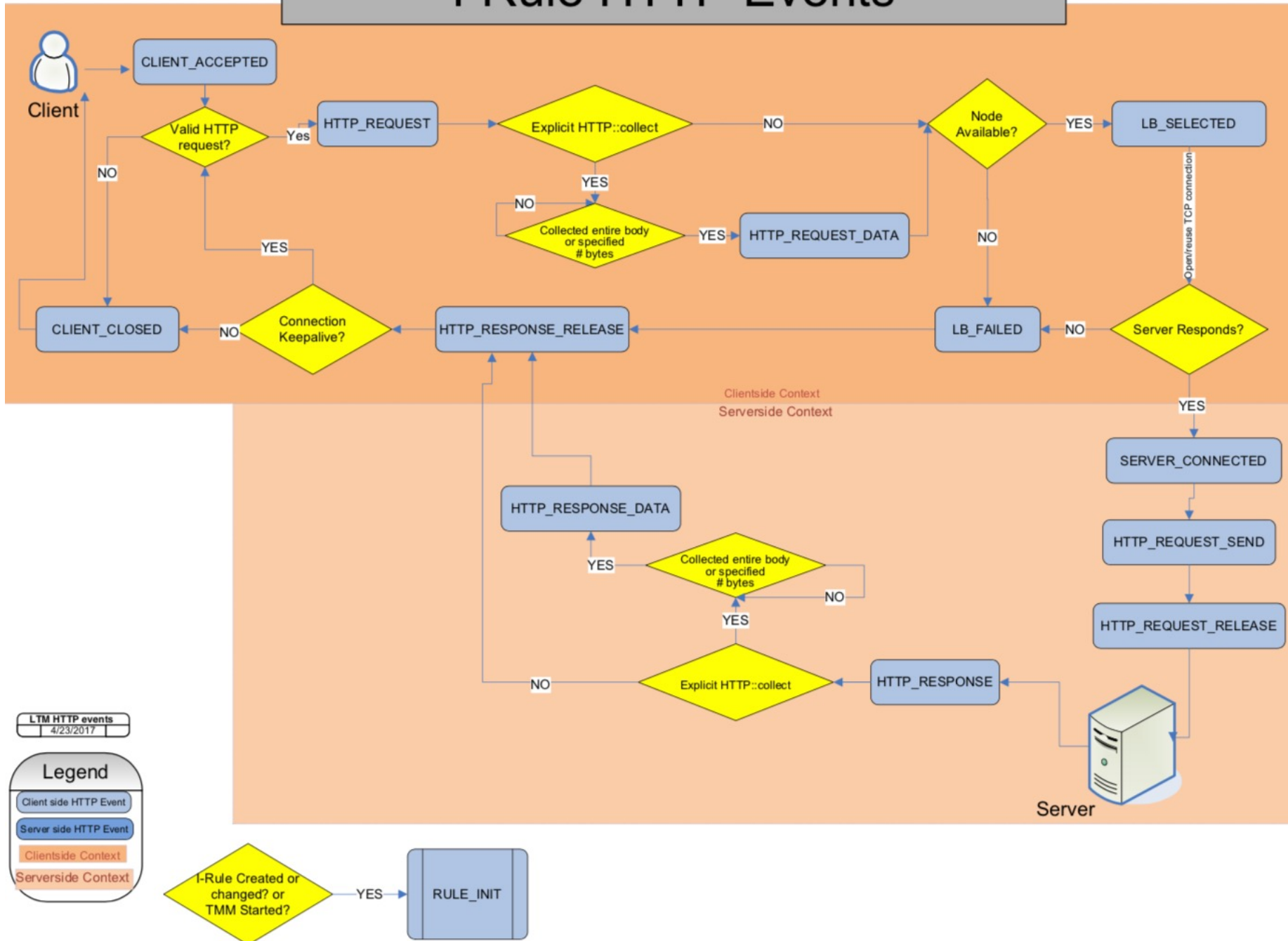


Connection
made?

Yes



I-Rule HTTP Events



Example: the event life cycle of an HTTP dialog



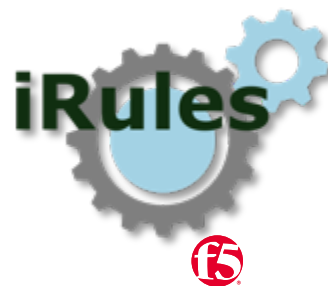
Client



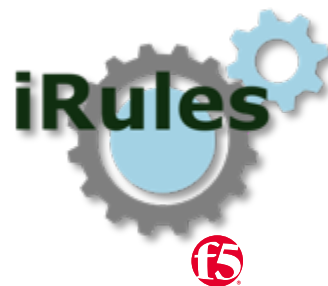
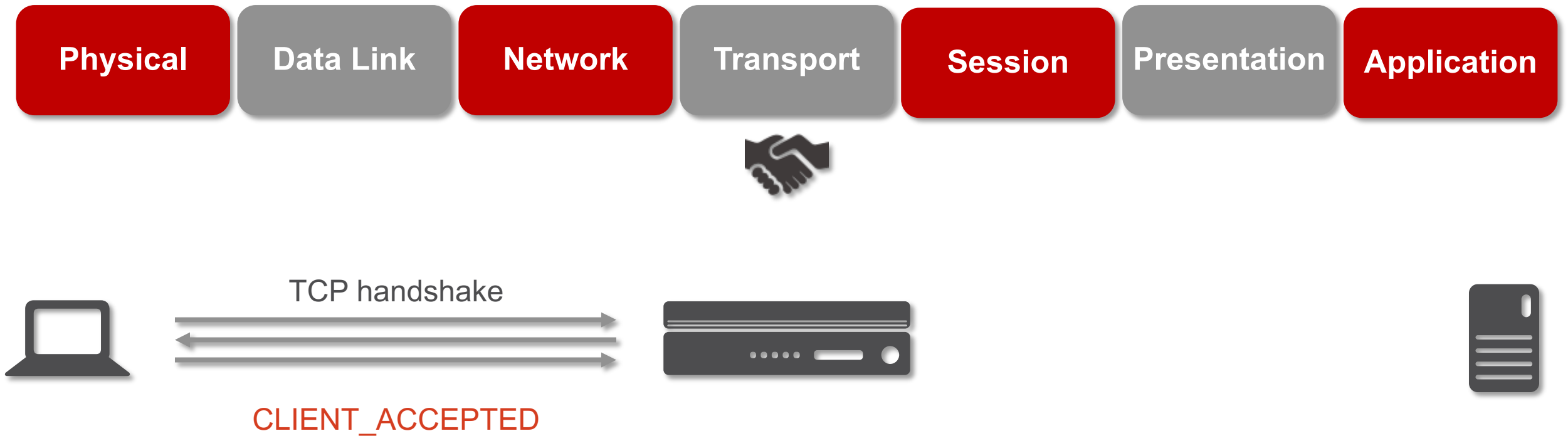
BIG-IP



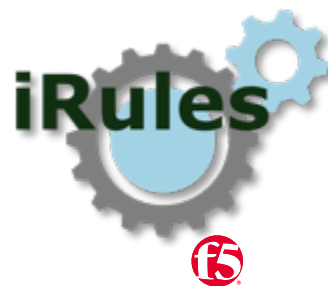
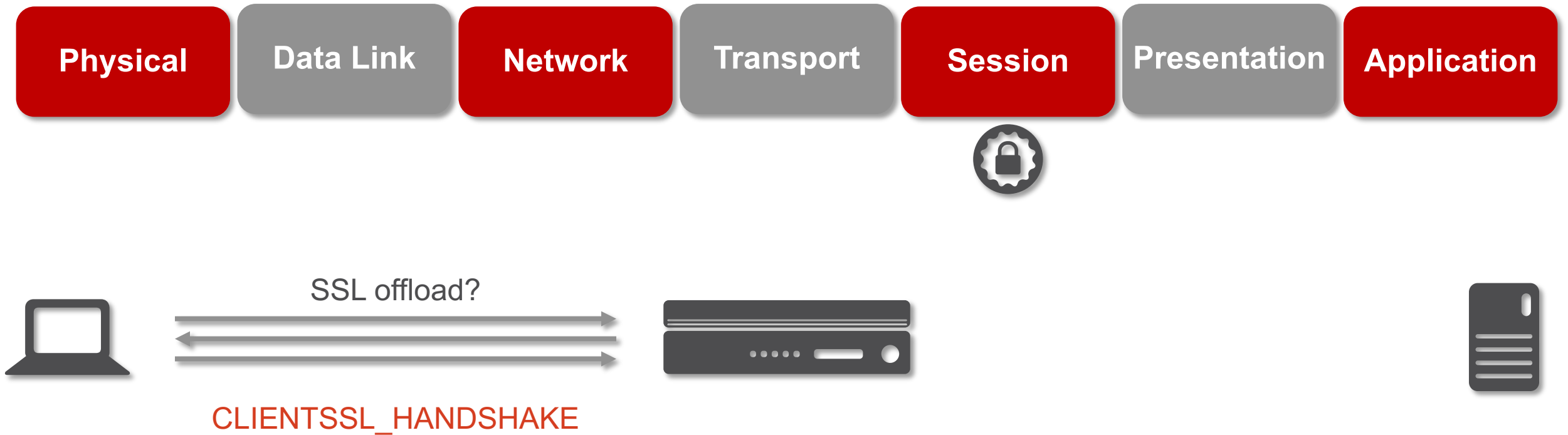
Server



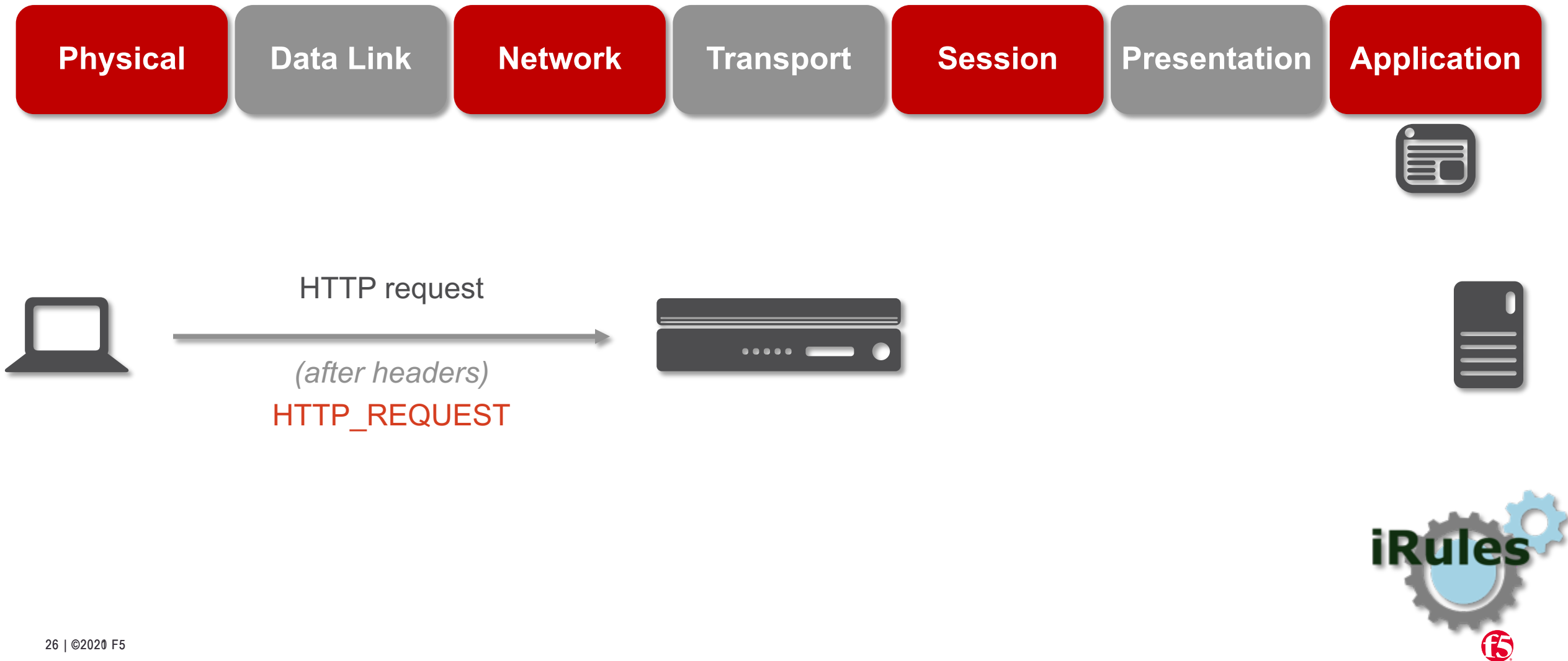
Example: the event life cycle of an HTTP dialog



Example: the event life cycle of an HTTP dialog



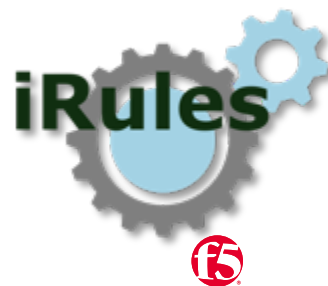
Example: the event life cycle of an HTTP dialog



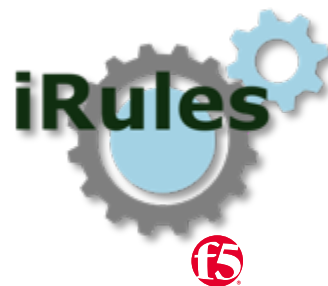
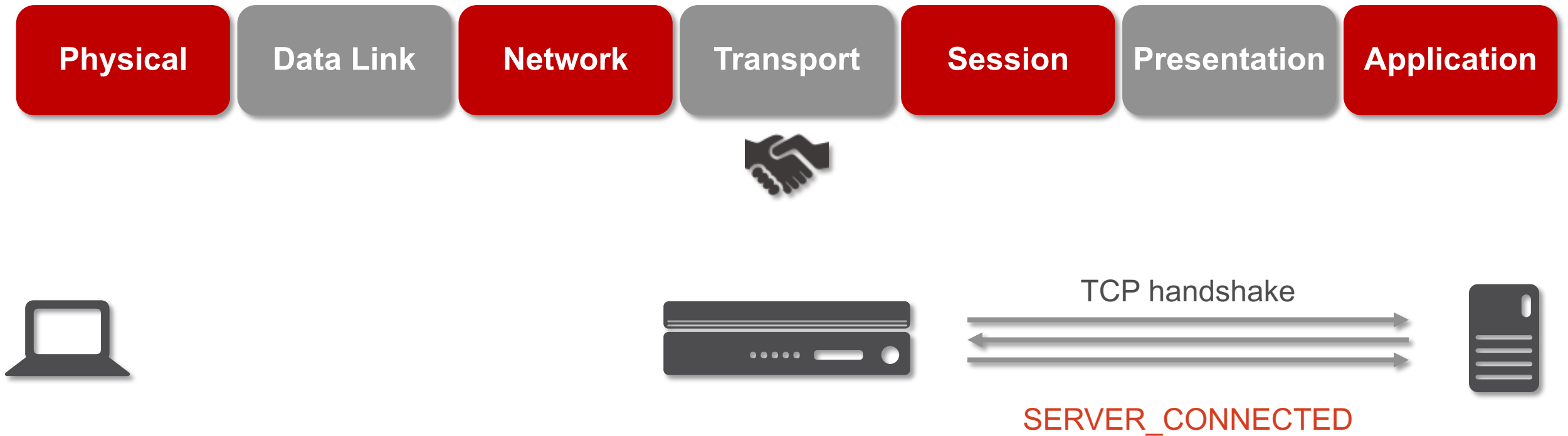
Example: the event life cycle of an HTTP dialog



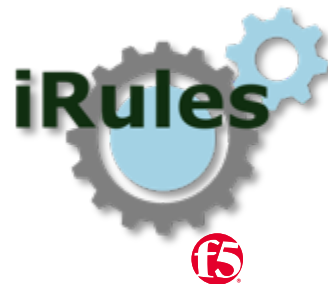
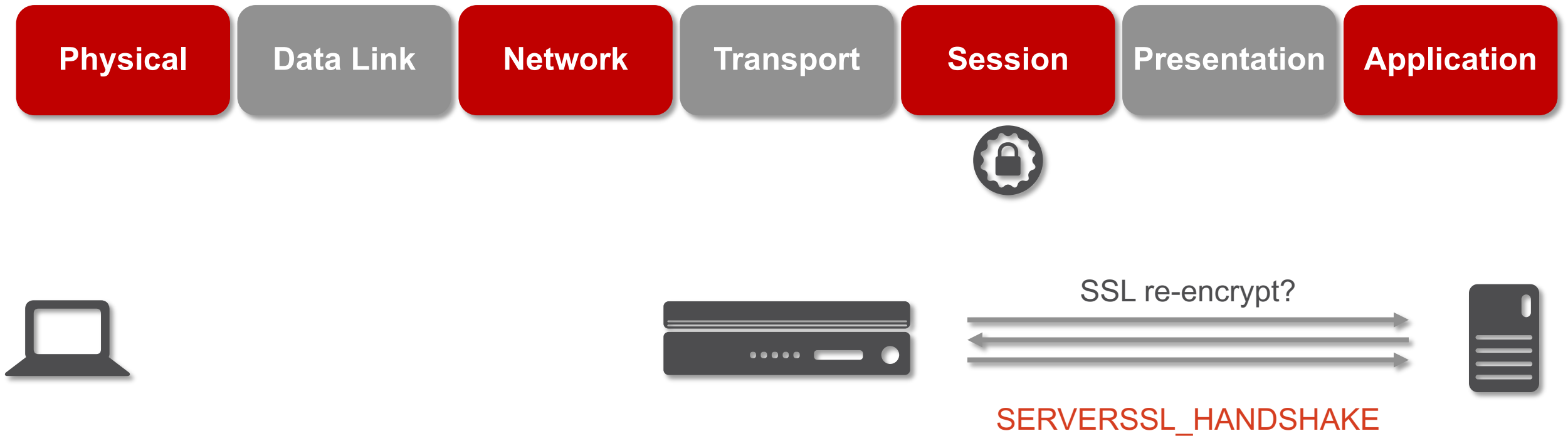
LB_SELECTED
LB_FAILED



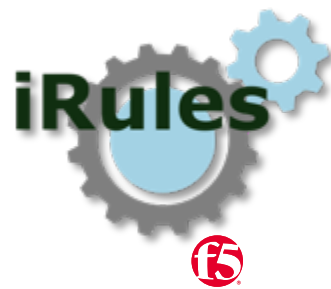
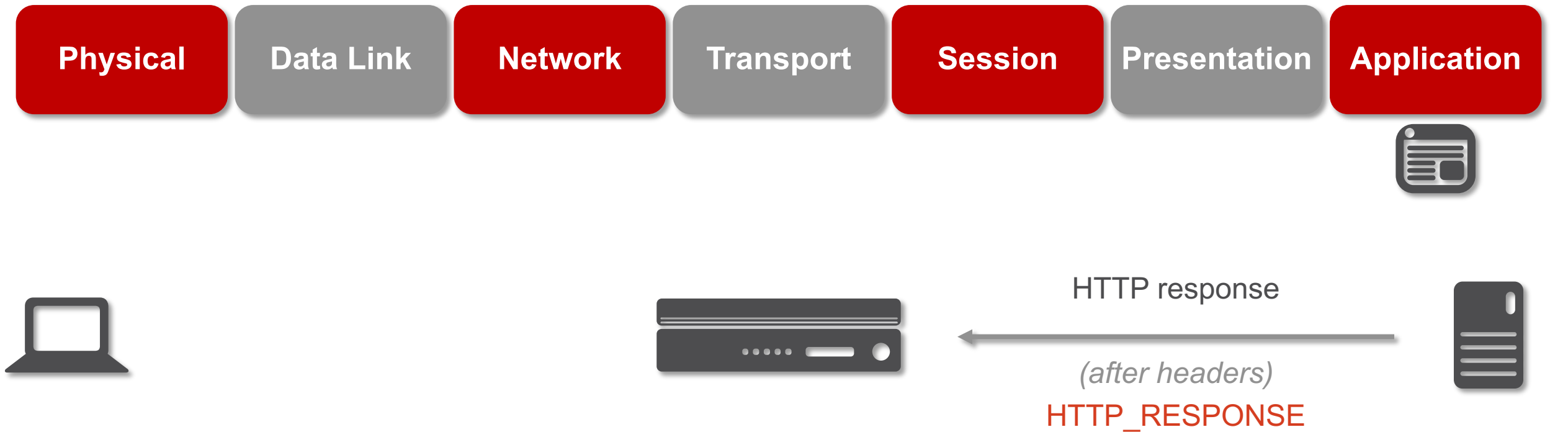
Example: the event life cycle of an HTTP dialog



Example: the event life cycle of an HTTP dialog



Example: the event life cycle of an HTTP dialog



The HyperText Transfer Protocol: cookies

- A cookie is a piece of text data sent by a server to a client to associate information with that client. When the server sets the cookie in a response, it specifies a name/value pair, and optionally a set of **scope** parameters:

```
Set-Cookie: myck=1; path=/; domain=y.com; secure; httponly  
Set-Cookie: myck=1; expires=Thu, 01 Jan 1970 00:00:00 GMT
```

- Once set by the server, and until it either expires or is removed, the client will send the cookie back to the server on **every request**.

```
Cookie: myck=1
```



The HyperText Transfer Protocol: cookies

- Cookie **scope** is defined by a set of optional parameters in the Set-Cookie header. These instruct the client on how and when to return the cookie.

```
path=/foo  
domain=f5.com  
secure  
httponly  
expires=Thu, 01 Jan 1970 00:00:00 GMT
```



iRules HTTP commands

- By far, one of the BIG-IP's greatest assets is its extremely rich set of events and tools to handle HTTP traffic. There are 10 events and 26 HTTP-specific commands. Starting with **request** events:

HTTP_REQUEST – triggered when the system fully parses the complete client HTTP request headers

HTTP_REQUEST_DATA – triggered when an HTTP::collect command has collected the specified amount of data

HTTP_REQUEST_SEND – triggered immediately before an HTTP request is sent to the server side TCP stack



iRules HTTP commands

- Then there are the **response** events:

HTTP_RESPONSE – triggered when the system parses all of the response status and header lines from the server response

HTTP_RESPONSE_DATA – triggered when an HTTP::collect command has collected the specified amount of data



iRules HTTP commands

- And since there are so many HTTP commands, we'll just look at a few examples. **HTTP::header** can read and write HTTP header data.

```
when HTTP_REQUEST {  
    if { [HTTP::header Host] starts_with "foo" } {  
        pool foo_pool  
    }  
}  
when HTTP_RESPONSE {  
    foreach aHeader { Server X-Powered-By } {  
        HTTP::header remove $aHeader  
    }  
    HTTP::header insert X-Local-Port \  
        [clientside [IP::local_port]]  
}
```



iRules HTTP commands

- Be mindful of the context in which a command is used. For example, the **HTTP::header** command used in requests and responses will produce different results.

```
when HTTP_REQUEST {  
    set sessionid [HTTP::cookie "appid"]  
    HTTP::header insert "X-Session-Id" $sessionid  
}  
when HTTP_RESPONSE {  
    if { $sessionid eq "" } {  
        if { [HTTP::header exists "X-Session-Id"] } {  
            HTTP::cookie insert X-Session-Id \  
                [HTTP::header X-Session-Id]  
        }  
    }  
}
```



iRules HTTP commands

- The ***HTTP::redirect*** command redirects an HTTP request or response to the specified URL.

```
when HTTP_REQUEST {  
    if { [HTTP::uri] starts_with "/auth" } {  
        HTTP::redirect "https://auth.y.com"  
    } elseif { [HTTP::uri] starts_with "/images" } {  
        HTTP::uri "/imagesrv"  
        pool image_pool  
    }  
}
```



iRules HTTP commands

- The ***HTTP::cookie*** command can read and write HTTP cookie data.

```
when HTTP_REQUEST {  
    if { [HTTP::cookie exists "oldcookie"] } {  
        set cookievalue [HTTP::cookie value "oldcookie"]  
        HTTP::cookie insert name "newcookie" value $cookievalue  
        HTTP::cookie remove "oldcookie"  
    }  
}
```



iRules HTTP commands: getting in the game

- Because everything else wasn't cool enough, we also have the ***HTTP::respond*** command, which lets you respond to a client request *on behalf* of the server. Some possibilities include:

Static HTML

JavaScript/CSS

Dynamic information

More flexible redirects

Adding/removing application functionality

Enhanced and extremely flexible authentication

No need for an actual server!



iRules HTTP commands: getting in the game

- Let's look at an example.

```
when HTTP_REQUEST {  
  if { [active_members mypool] < 1 } {  
    switch [string tolower [HTTP::uri]] {  
      "/logo.png" {  
        HTTP::respond 200 content [ifile get logo] "Content-Type" "image/png"  
      }  
      default {  
        HTTP::respond 400 content "...maintenance page HTML..."  
      }  
    }  
  }  
}
```

HTTP::respond 200 content {<html>...</html>} "header name" "header value"

code keyword content optional headers



DevCentral **is...**

Resources

Wiki-based access to F5
product and API
documentation.

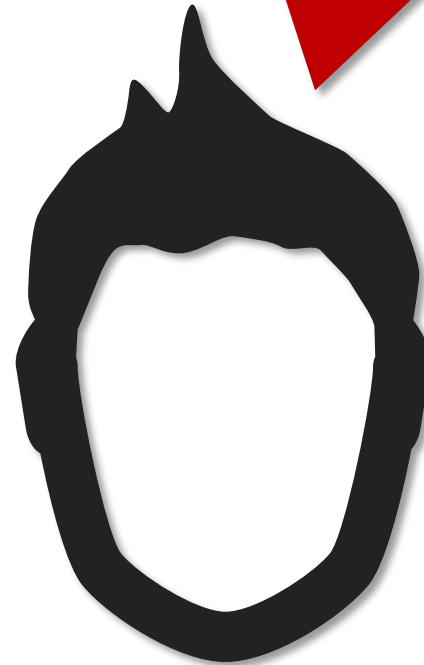
<https://clouddocs.f5.com>

Forums

A place to ask and answer
F5 product and API
technology questions.

150,000+

and growing community.



<https://devcentral.f5.com>

