Application Layer Protocol Negotiation

TLS extension for application layer protocol negotiation within the TLS handshake

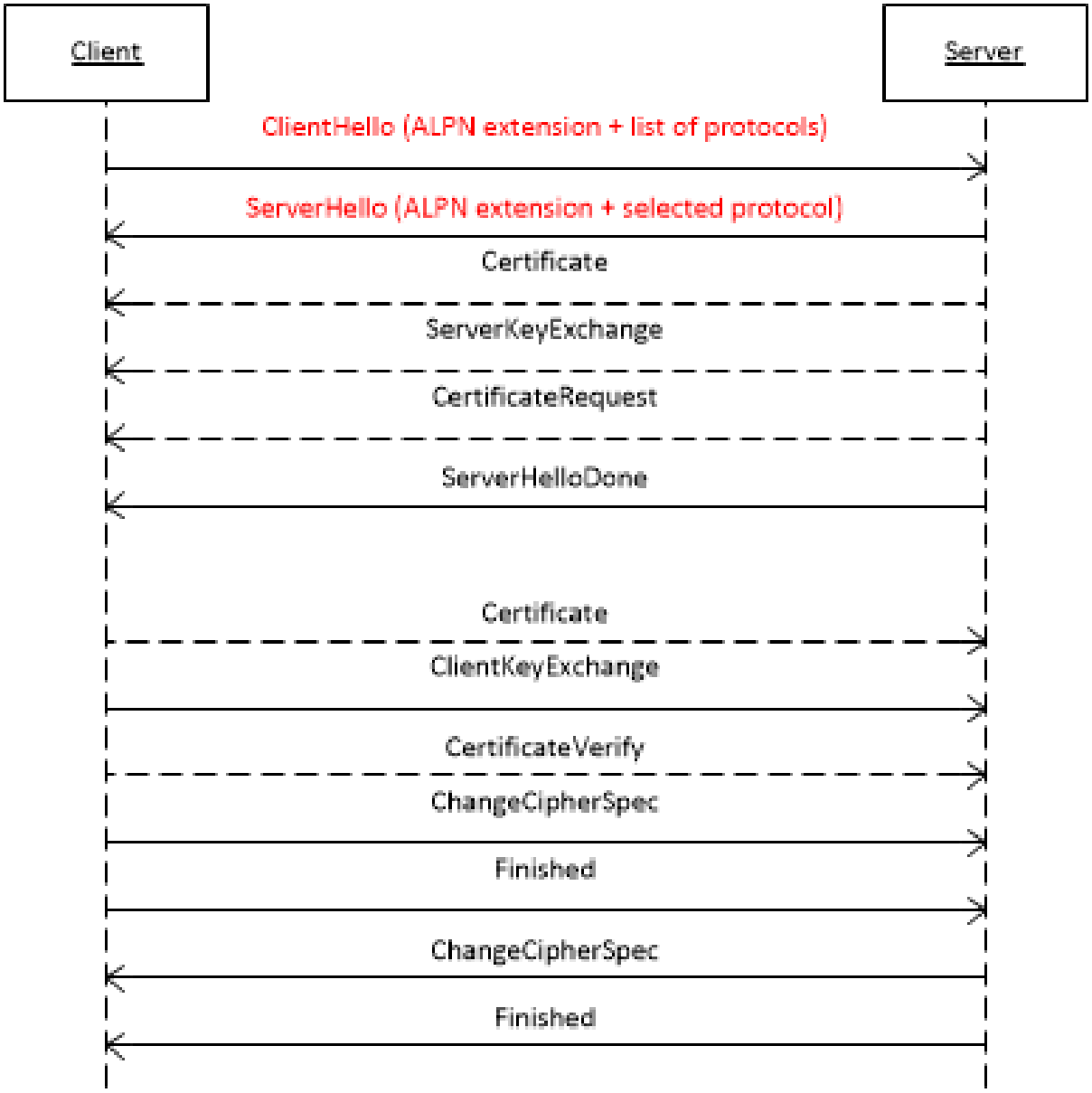
Background and Design Goals

HTTPBis WG requested TLS support for negotiating application layer protocols such as HTTP 1.1 and HTTP 2.0.

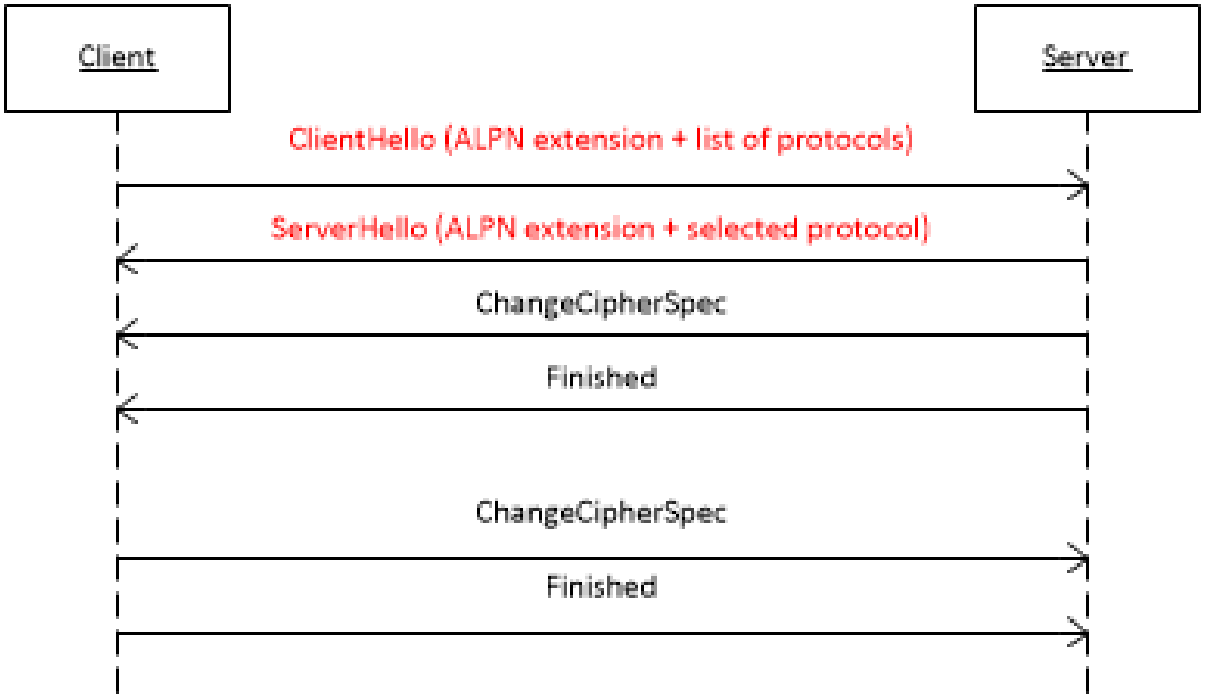
Design goals:

- Negotiate application layer protocol for the connection.
- Minimize connection latency.
- Align with existing TLS extensions.

Full TLS Handshake with ALPN



Abbreviated TLS Handshake with ALPN



ALPN Extension Structure

- The "extension_data" field of the ALPN extension SHALL contain a "ProtocolNameList" value.

```
opaque ProtocolName<1..2^8-1>;
struct {
        ProtocolName protocol_name_list<2..2^16-1>
} ProtocolNameList;
```

- When sent with the ClientHello message, "ProtocolNameList" contains the list of protocols advertised by the client, in descending order of preference.
- When sent with the ServerHello message, "ProtocolNameList" MUST contain exactly one "ProtocolName“ representing the selected protocol.

Protocol IDs and Protocol Selection

- Protocols IDs are IANA registered, opaque, non-empty byte strings.
- Initial registrations have been requested for HTTP/1.1, SPDY/1, SPDY/2, SPDY/3.
- If the server supports no protocols that the client advertises, the server SHALL respond with a fatal "no_application_protocol" alert.

ALPN Design Considerations

- Protocol selection on the server allows certificate to be chosen based on the negotiated protocol.
- The negotiated protocol is known after the first network roundtrip.
- The "extension_data" field of the ALPN extension allows re-use of the existing parsers.
- TLS renegotiation can be used to negotiate an application protocol with confidentiality.

Changes Since IETF88

- Minor re-wording in sections 3.2 and 4 to clarify that the application protocol can be renegotiated in the course of TLS session renegotiation.
- Added text in section 5 “Security Considerations” to highlight the risks of sending sensitive protocol IDs in the clear.
- IESG has evaluated and approved the ALPN draft.
- Minor updates resulting from IESG review will be incorporated once the I-D submission tool is reopened.

Available Implementations & Tools

- New since IETF88: NGINIX added ALPN support.
- ALPN is implemented in several HTTP/2 prototypes, including Katana, Mozilla, Chromium, iij-http2, GFE.
- ALPN patch for OpenSSL contributed by Google.
- ALPN support for Wireshark network analyzer contributed by Akamai.

ALPN Deployment

- *.google.com servers have ALPN enabled.
- Google Chrome and IE11 support application protocol negotiation via ALPN.
- F5/BIG-IP FW versions older than 10.2.4 cannot handle ClientHello messages longer than 255 and shorter than 512 bytes. This is a general issue e.g. when adding cipher suites, extensions, or using SNI with a long server name. The use of ALPN extension can also expose this bug.
- A workaround for the F5 issue exists: ClientHello padding TLS extension (draft-agl-tls-padding). Early IANA code point allocation for this extension has been requested.

Links and Contact Information

- ALPN Draft: http://datatracker.ietf.org/doc/draft-ietf-tls-applayerprotoneg
- ClientHello padding TLS extension to work around the F5 issue: https://datatracker.ietf.org/doc/draft-agl-tls-padding/

- Stephan Friedl sfriedl@cisco.com
- Andrei Popov andreipo@microsoft.com
- Adam Langley agl@google.com
- Emile Stephan emile.stephan@orange.com