HTTP as the Narrow Waist of the Future Internet

Piotr Leszczyński

11.01.2012

Motivation

- A lot of research on new Internet architectures during few last years. Many proposals to address the shortcomings of IP protocol in terms of flexibility, scale and security.
- Explosive growth of HTTP traffic since the advent of the web driven by the prevalence of the existing HTTP infrastructure (like CDNs, HTTP proxies and caches), the ease of deploying new functionality on the data path via reverse and forward proxies, and the ability of HTTP to penetrate corporate firewalls.

Introduction

The goal of this presentation is to evaluate HTTP with respect to the existing Internet proposals and answer the following questions:

- What are the properties aimed by these architectures that HTTP already provides, and what are the ones it does not?
- What are HTTP's main drawbacks?
- Can these drawbacks be addressed by extending HTTP, or are they the result of fundamental limitations of HTTP?

List of contents

HTTP takes over the world

- Since mid-90's HTTP became the dominant traffic in the internet because of the emergence of the web.
- After next few years video and audio traffic seemed to challenge HTTP's dominance.
- However because of having a lot of desired features (like huge infrastructure) HTTP appeared to be a good base to streaming so now it is still the most popular protocol in the Internet.

HTTP chunking

HTTP chunking enables the delivery of video and audio over HTTP. The basic idea is to chunk a video stream into blocks of a few seconds each, and then distribute these blocks as individual files.

Such approach has the following advantages over traditional streaming protocols:

- It increases distribution scale by using HTTP caching proxies and reduces costs because HTTP servers are typically open-source, unlike the streaming servers.
- Availability improvement: if an HTTP server fails, the client can request subsequent chunk form another server.
- Quality improvement: client can request multiple chunks simultaneously.
- Penetration improvement: HTTP usually isn't blocked by firewalls.

HTTP vs the brave new world

For the clarity of the comparison, let's classify research proposals to improve the Internet architecture into five categories:

1. Transforming the Internet into a content centric network.
2. Enabling the explicit use of middleboxes.
3. Introducing more flexible communication patterns, like mobility, anycast and multicast.
4. Increasing network security.
5. Extending the Internet routing policies and adding QoS policies.

1. Content centric networks

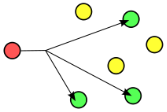
- HTTP is already a content centric protocol, as HTTP requests deal primarily with retrieving, storing, and updating content.
- This HTTP's feature is strongly supported by massive caching infrastructure.
- Important difference between HTTP and research proposals lays in the way of content naming. HTTP forces to use DNS names as a part of the content names. If more sophisticated mechanism of naming is necessary HTTP might not be appropriate.

2. Explicit middlebox support

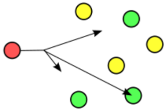
- In general putting middlebox in IP layer is difficult, because it doesn't expose a middlebox-like abstraction to end-hosts. Any way to do it is to place middlebox physically on IP data path.
- In contrast HTTP already provides support for middleboxes via explicit forward proxies and via reverse proxies.

3. Flexible communication

- Idea is to add such functionalities like mobilty, multicast, anycast, multi-path and delay tolerant networks.
- Aforementioned improvements are very desired in the current Internet.
- HTTP partially offer some of them: e.g. single-source multicasting by caching and anycasting by DNS anycast functionality.



Multicast



Anycast

- However HTTP doesn't support simultaneous end-host mobility or multi-path communications.

4. Security

- HTTP deosn't have any protection against DoS attacks.
- But has some support to protect against impersonation or eavesdropping (HTTPS).

5. Routing and QoS

- Many proposals have aimed to improve the robustness, efficiency, and security of inter/intra domain routing, as wall as to provide QoS guarantees.
- In general HTTP doesn't address any of these challenges.

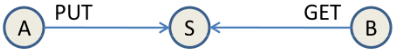
Comparison summary

Property	**HTTP support**
Content centric network	Yes, via named resources and caching
Middlebox support	Yes, via proxies
Additional communication patterns like mobility, multicast, anycast, multipath/multihomed	Yes, via proposed S-GET extension, caching, CDNs and DNS
Security extensions	Yes/Partial via proposed S-GET extension, HTTPS and adding authentication field in header
Routing policy extension	No, but can be implemented mostly independently of HTTP

Datagram services over HTTP - motivation

- **Goal**: create a communication model on top of HTTP to exchange application data units (ADUs) between two or more clients.
- **Additional goal**: low latency of such connection
- The examples of use are live video streaming and VoIP.

Datagram services over HTTP - natural approach

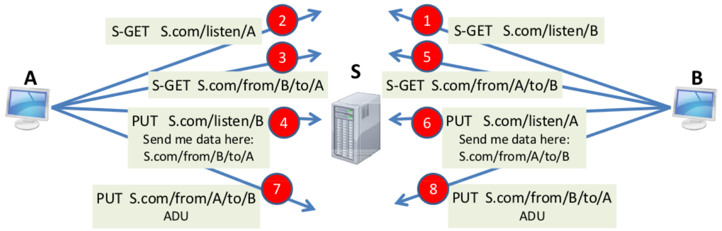


- `A` publishes data to HTTP server `S`
- `B` receives the piece of data from server `S`
- Problem lays in HTTP's pull abstraction. Client `B` doesn't know when new data has became available. The only way to reduce latency is to periodically check the data availability. However check frequency high enough to achieve desired latency level generates too much traffic.
- **Solution**: adding a push abstraction to HTTP.

Datagram services over HTTP - S-GET

- Push abstraction will be carried out by new type of get request, calles S-GET (*Subscribe-GET*).
- S-GET is exactly the same as GET with the difference that response doesn't come immediately. Client that sent S-GET request doesn't get reponse until:
  - Server gets new PUT request on content connected with S-GET **or**
  - Timeout has reached

Datagram services over HTTP - communication example



Idea: exchanging ADUs between `A` and `B`.

Datagram services over HTTP - benefits

- Full mobility of all clients (until server doesn't change the location).
- Multicasting achieved by registering S-GET on the same content by many clients.
- Multiple homed clients can setup different S-GET receive channels for each interface.
- Multipath by using many servers.
- Good firewall penetration because of using HTTP.

Datagram services over HTTP - other considerations

- Client has to refresh S-GETs if it wants to wait constantly.
- Brilliant way to use caches: caching S-GET requests instead of responses.
- Independence from transport layer. S-GET could be based on non-tcp protocol.

Datagram services over HTTP - evaluation

- **Latency**:
  - extra network latency due to using an off-path server as relay (outside of the scope)
  - HTTP processing overhead at end-point and the indirection server (negligible)
  - delay introduced by the ”store and forward” machanism (negligible for small blocks; not important for bigger)
- **Throughput**: not significent difference
- **ADU Size Overhead**: about 8% of block size (might be essential in some cases)

The end

Questions?