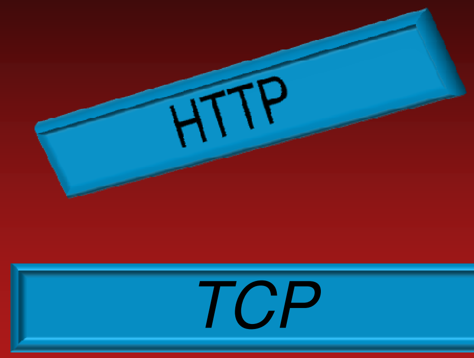
Turning HTTP into a standalone layer in the stack

Decoupling HTTP from TCP



Lloyd Wood
Cisco Systems

TSV Area session
IETF 75, Stockholm, July 2009.

Decoupling HTTP from TCP

- HTTP has always run over TCP
- Desire to run HTTP over other transports, e.g.:

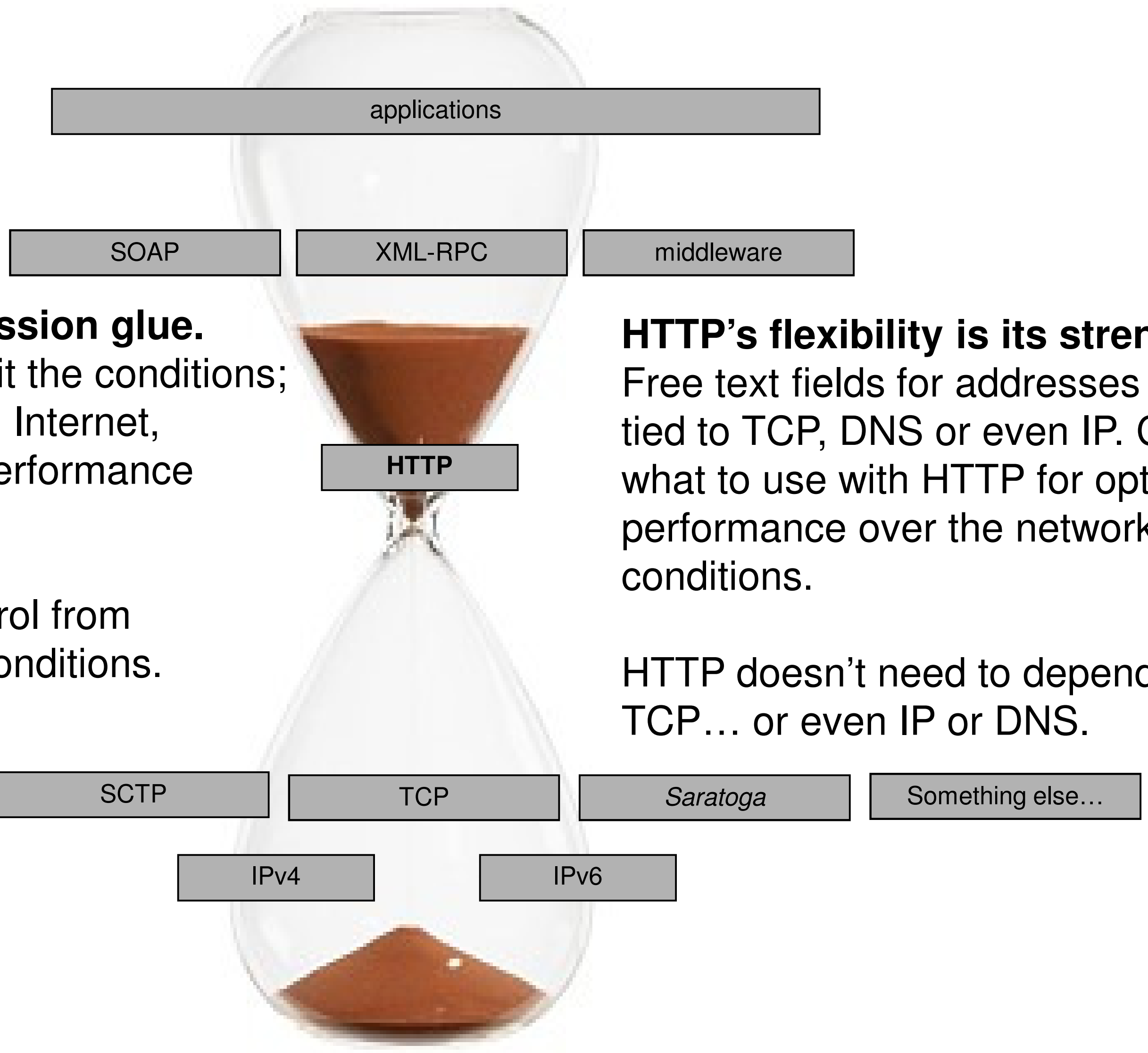
  SCTP, which may eventually completely replace TCP! Really!

  Transports for unusual environments, e.g. dedicated/errored links where applications that speak HTTP are still useful, but TCP is less so thanks to its operating assumptions (*Saratoga* fits here).

  Even direct over something like HDLC…

- HTTP needs to be decoupled from TCP.
- Divide HTTP from TCP to make a true(ish) session layer.
- What HTTP requires from transport isn't that onerous – a bidirectional bitstream, basically.
- But should HTTP take advantage of features offered by various transport protocols, and if so, how?

HTTP is the new waist in the hourglass



HTTP is the universal session glue.
choose the transport to suit the conditions; TCP or SCTP in traditional Internet, something else for high performance across dedicated links.

Separate the session control from transport, link and traffic conditions.

HTTP's flexibility is its strength
Free text fields for addresses aren't tied to TCP, DNS or even IP. Choose what to use with HTTP for optimum performance over the network conditions.

HTTP doesn't need to depend on TCP… or even IP or DNS.

Various drafts looking at aspects of this

- Putting HTTP over SCTP:

  draft-natarajan-http-over-sctp

- And making it coexist with TCP for web browsers:

  draft-wing-http-new-tech (HTTP for human web use)

- Carrying HTTP over any locally-appropriate transport:

  draft-wood-dtnrg-http-dtn-delivery (HTTP for machines)

- And specifying what transport or port to use, dynamically or statically:

  draft-wood-tae-specifying-uri-transports

  draft-jennings-http-srv

  NAPTR etc.

Why is this of interest to transport area?

- *httpbis* group is currently chartered to rewrite HTTP/1.1 specification; could call out (the few) TCP-specific dependencies.
- But how HTTP might be carried over different transports requires transport-area expertise.
- Remember HTTP/0.9 – opening/closing TCP for single file transfer led to poor performance, but took *years* to fix.
- Does HTTP just use all transports as dumb TCP-like bitstreams, or does it take advantage of available functionality (e.g. SCTP streams, unordered delivery…)? Expertise is needed.

Recommendations

- Remember that there are different use cases – HTTP for humans (trying different transports, seeing what works) and HTTP for machines (explicit configuration, custom controlled environments).
- Transport area should take an interest in how various transports can carry HTTP.
- If transport area doesn’t, it will have to eventually – remember the early days of HTTP with TCP!

Questions?
thankyou