Moving data in DTNs with HTTP and MIME

Making use of HTTP
for delay- and disruption-tolerant networks
with convergence layers

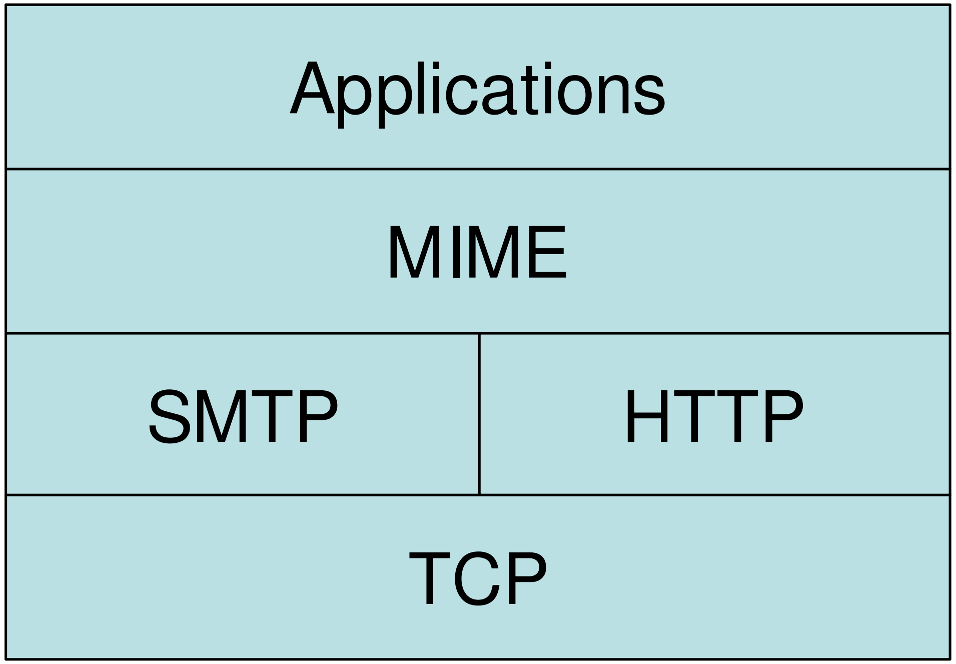
Lloyd Wood, Daniel Floreani, Peter Holliday, **Ioannis Psaras**

e-DTN workshop, ICUMT, St Petersburg, 14 October 2009.

Why use HTTP?

- MIME describes the things we move around the network. The most successful protocols support MIME.

- HTTP is the simplest MIME wrapper.
- HTTP provides infinitely-flexible text metadata.



Applications

MIME

SMTP

HTTP

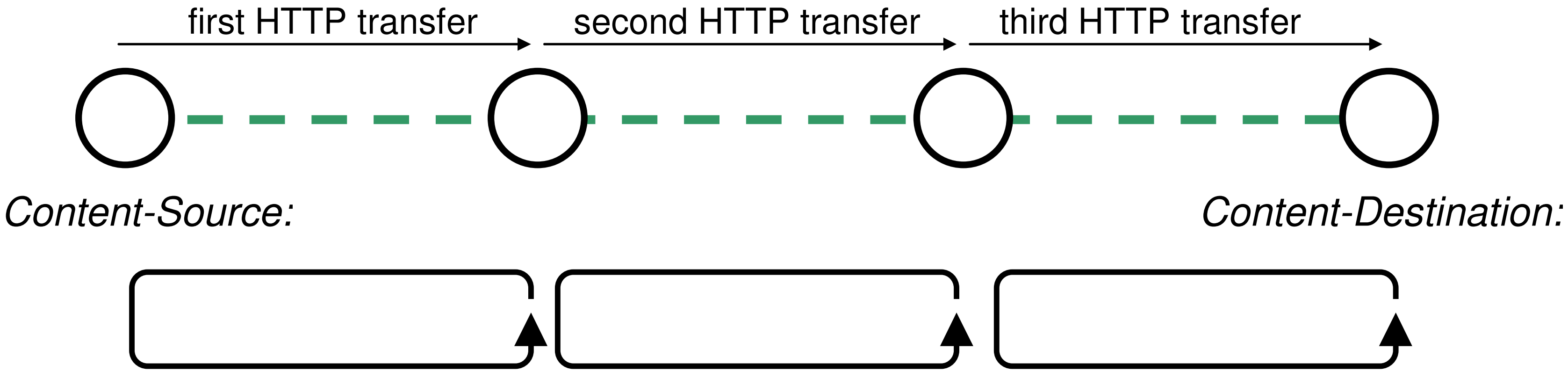
TCP

Decoupling HTTP from TCP underway

- Proposal in IETF to use HTTP over SCTP.
- Could use HTTP over anything giving a reliable bitstream – HDLC, *Saratoga*, even direct over CCSDS bitstream service.
- Makes HTTP useful in more environments. **Makes HTTP a standalone layer in its own right.**
- Decoupling HTTP from TCP opens doors to convergence layers for HTTP and to HTTP-DTN.

HTTP (not the web) transports MIME

- Use HTTP hop-by-hop between neighbouring DTN nodes.



- Allow HTTP to be run over different transports: TCP, SCTP, *Saratoga*… HTTP can be separated from TCP's limitations. Divide HTTP from transport to make a true session layer.
- Adapts HTTP to each local environment.

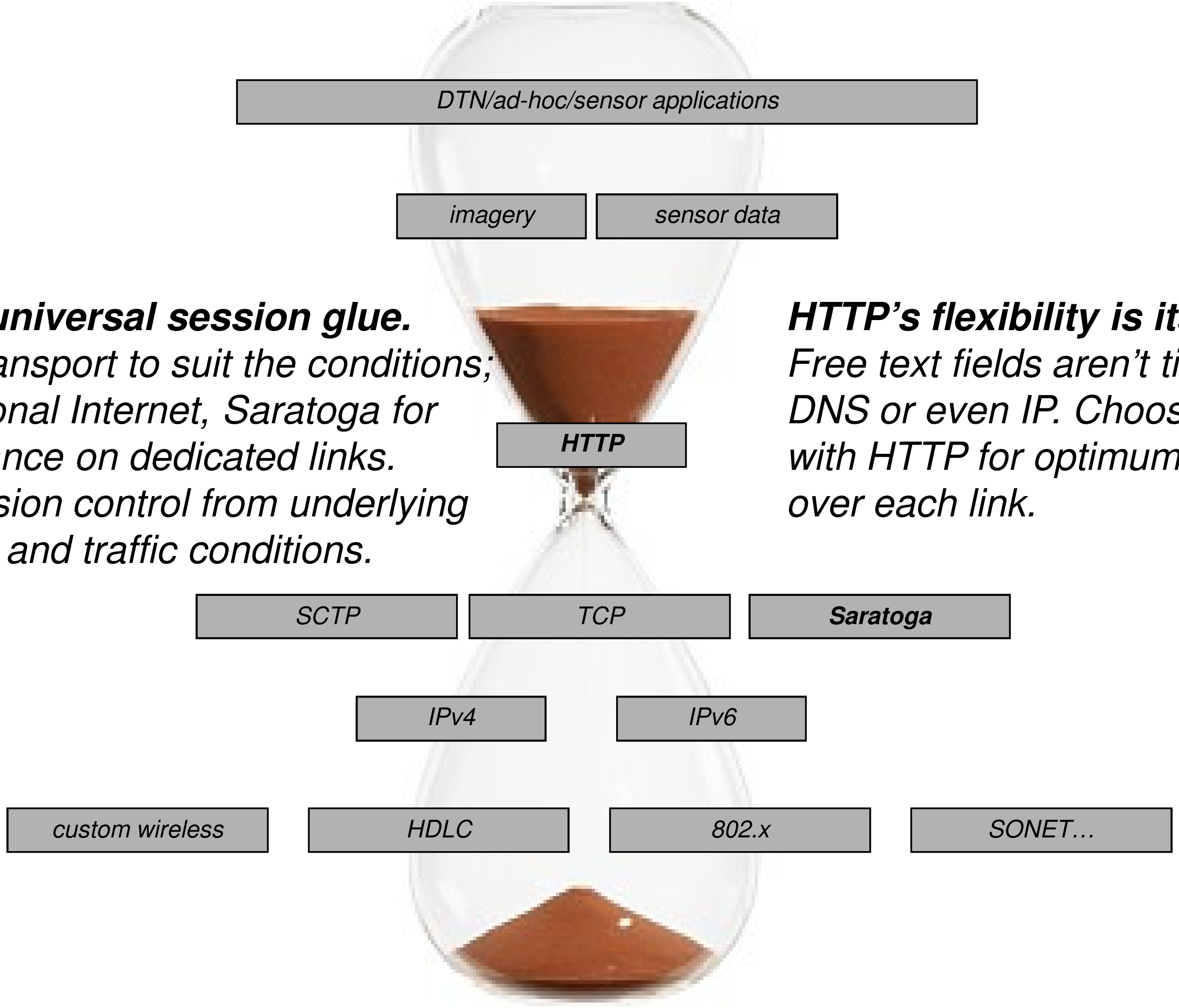
What makes HTTP*-DTN* special?

- Two new **Content-*** headers:

  Content-Source: where the object is originally from

  Content-Destination: final destination
- Basic HTTP rule: **Content-*** headers are special . If Content-*blah* is unfamiliar, reject the transfer.
- This makes HTTP*-DTN* separate from, and not polluting, existing web. Unlikely to alarm W3C.
- Optional e2e reliability over payloads by reusing existing **Content-MD5:** header or similar.
- Header/metadata reliability a bit trickier – may need new headers. HTTP already supports ‘per hop’ limited-scope headers.
- New Package- headers can *package* related objects together, track if they’ve all arrived or not.

HTTP is the waist in *this* hourglass



DTN/ad-hoc/sensor applications

imagery | sensor data

HTTP

SCTP | TCP | **Saratoga**

IPv4 | IPv6

custom wireless | HDLC | 802.x | SONET…

HTTP is the universal session glue.
choose the transport to suit the conditions; TCP in traditional Internet, Saratoga for high performance on dedicated links. Separate session control from underlying transport, link and traffic conditions.

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

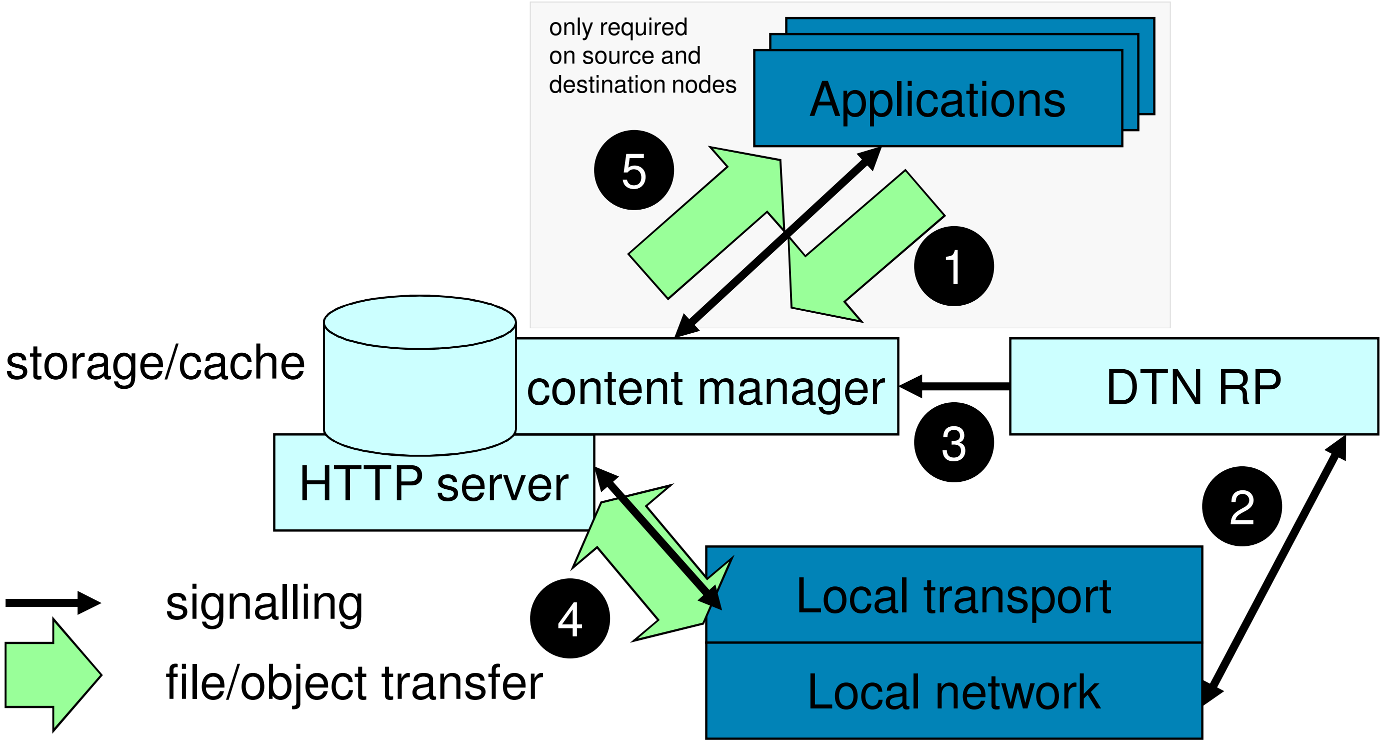
HTTP-*DTN* advantages

- Text fields aren’t tied to IP, TCP or to DNS. Could implement HTTP over own stack, with own routing namespace, etc. Easily modifiable, not a strange binary format.
- Doesn’t require a two-way session; HTTP PUT could be entirely unidirectional.
- Reuses large body of existing code and well-understood functionality. Only minor changes.
- Possible to build on top of HTTP-*DTN* base to reuse pieces of web infrastructure, e.g. SOAP.
- Shares some of the Bundle Protocol’s problems, e.g. universal clock, but gets there with far less development work. Very *very* simple.

What model do we use with HTTP DTN?

- We don’t have to even use IP, but…
- **We still believe IP is useful for operational use of delay/disruption tolerant networks** – IP is not *just* convenient/cheap for prototyping DTN code.
- Make each transport layer work with HTTP and IP. The transport between HTTP and IP must support HTTP’s simple session semantics.
- Pick the transport to match the local environment.
- How do we build these transfers into a bigger architecture that can make forwarding and routing decisions? Open – there are many pieces of IP-based infrastructure that *may* be reusable, depending on the exact scenario.
- Early days, interesting adaptation questions to address.

A potential HTTP-DTN node



Issues

- Security

  Could reuse https: for hop-by-hop security.

  Could use S/MIME for end-to-end security – or applications could implement their own. Unsure. Early days yet.

- Timestamps

  pretty much the same timing/sync issues as the Bundle Protocol has come across.

- Header overhead

  may be significant for small transfers; it’s the cost of flexibility. (Bit efficiency was *gopher*’s strong point.)

Questions?
Thank you