Application Protocol Design Considerations for a Mobile Internet

2006-12-01 MobiArch Workshop 2006

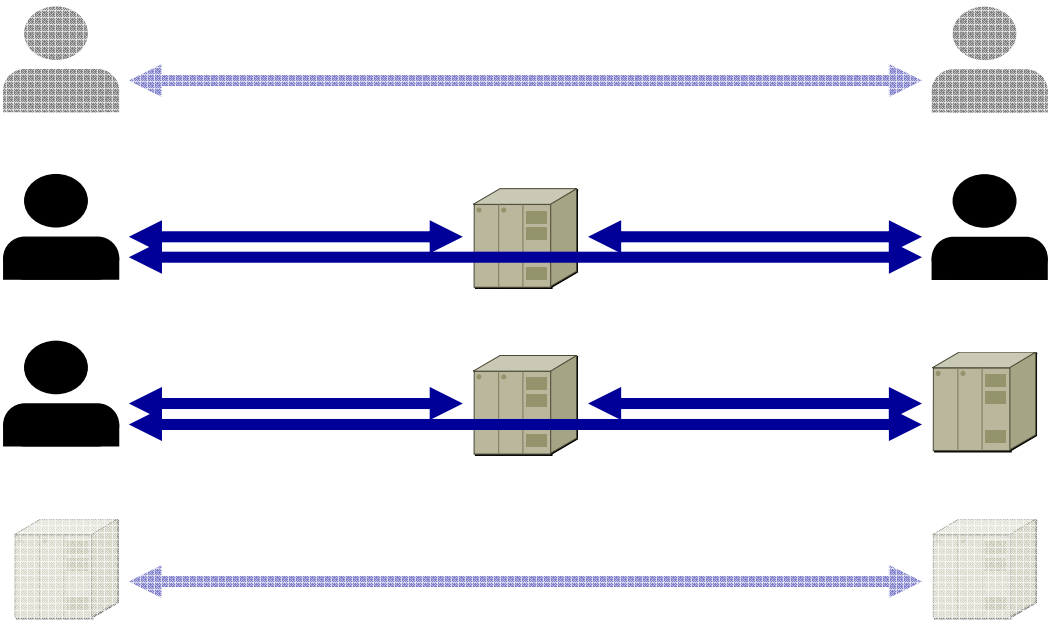
Jörg Ott jo@netlab.tkk.fi

Mobile and Nomadic Users

- Mobility is increasingly dominating Internet usage
  - Always best connected: ubiquitous connectivity and seamless access
- Dealing with mobility
  - Link layer handovers (WLAN, GSM, UMTS) + optimizations
  - Mobile IP, HIP, cross-network and cross-provider optimizations
  - Transport layer enhancements
  - Introducing a “session layer”
  - (Little application-specific support)
- Mobility implies disconnection
  - No coverage
  - Economically infeasible (at least today)
  - Social, legal restrictions
- Connectivity is neither ubiquitous nor seamless
  - Abstractions become leaky

Debunking the Need for The Myth of Seamless Connectivity

Connectivity need not always be Seamless or Ubiquitous



Connectivity need not always be Seamless or Ubiquitous

- Many applications are asynchronous in nature
  - No need for “always on” connectivity today
  - Examples: email, file transfer, peer-to-peer, even presence and IM to some degree
- Applications don’t communicate most of the time
  - Users read, type, or do other things
  - Examples: web, email, calendar, chat, presence, …
- Users don’t have to perform “busy waiting”
  - Let the applications operate asynchronously and notify the user when done
  - Examples: (peer-to-peer) downloads, tabbed browsing, email, …
- Some “synchronous” applications may even fall back to asynchronous operation
  - Examples: voice mail, short and (not so) instant messaging, …

…regardless of whether mobile or not.

HELSINKI UNIVERISITY OF TECHNOLOGY
NETWORKING LABORATORY

Application semantics do not require permanent or “end-to-end” connectivity…

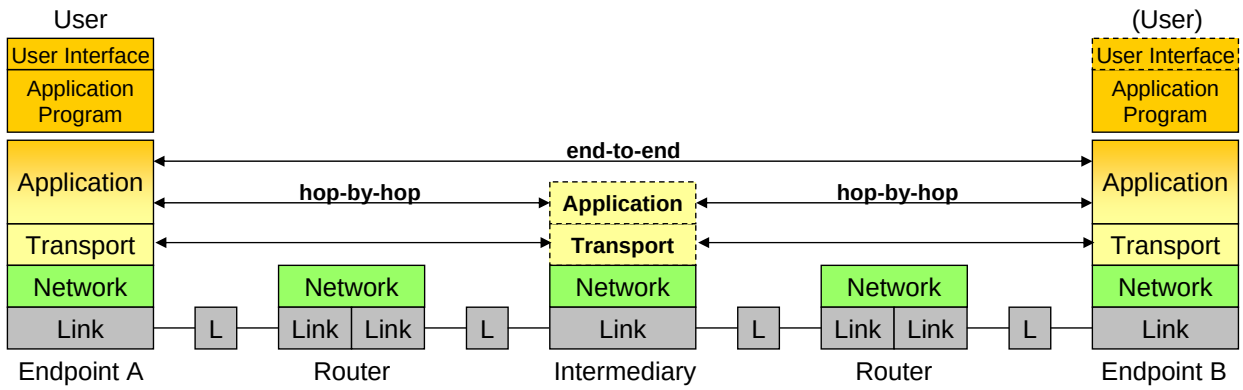
…but many application protocols do.

Application Protocols: Actors

- Endpoints
  - User agents, origin servers
- Intermediaries: notion depends on the application
  - Hidden vs. visible
  - Facilitating rendezvous
    - SIP servers, mail servers
  - Relaying / forwarding functions
    - Mail servers, SIP servers, web proxies (firewall traversal)
  - Necessary or useful application functions
    - Mail servers: storage, protocol conversion, virus checking, …
  - Optimization application functions
    - Web caches
  - Lower layer functions (hidden)
    - Firewalls, NATs, …

Application Protocols: Actors

- Intermediaries and Protocol Semantics
  - May become (single) points of failure
  - May hinder or limit communication
  - May break up end-to-end communication
    - limit application semantics to hop-by-hop
  - May require trust to perform some of their functions



Application Protocols: Actions (1)

- Chattiness
  - Many (end-to-end) interactions: build up and advance state step by step
  - Examples: SMTP, POP/IMAP, HTTP digest, TLS, SIP
  - Counter examples: S/MIME, email message bodies
- State often tied to underlying transport connection
  - Breaking the transport impacts the application, too
  - Limits semantics to the reach of the underlying transport
  - Examples: SMTP, POP/IMAP, FTP, TLS
  - Counter examples: HTTP, SIP, RTSP
- Timeouts
  - Degree of coupling between ends
  - Timeouts usually not adaptive
  - Hard to tell temporary unreachability from permanent failure
  - Examples: HTTP, SMTP

Application Protocols: Actions (2)

- Security: Often mapped to underlying security mechanisms
  - May limit reach of security to the next intermediary
  - Requires trust in intermediaries, possibly transitive trust
  - Security may break with underlying communication relationship
  - Examples: TLS, IPsec, tunneling
  - Counter examples: HTTP digest authentication, SIP, S/MIME
  - Security association setup may use highly interactive protocols, too
    - May depend on infrastructure (e.g., PKI, DNS)
- Infrastructure use: Naming & Addressing
  - IP addresses still used for identification
    - Moving towards URIs, DNS names, or other more stable identifiers
  - DNS lookup: indirection to resolve name into an IP address
    - Somewhat similar: Mobile IP home agent, HIP rendezvous server
  - Infrastructure must be reachable in the first place

Application Programs, (G)UIs & Users

- End-to-end semantics ultimately involve the user
- Tradeoff: persistence vs. responsiveness
  - Again: temporary unavailability may be indistinguishable from failure
  - It's all about timeouts and about user patience
  - How long to retry and when to declare failure
    - "Failures" may lead to manual retries by the user
  - Abstraction from underlying network prevents cues in the (G)UI
- Limited support for asynchronous (batched) operation
  - Examples: tabbed browsing, offline email
  - Still many manual interactions and retries required

Some design considerations…

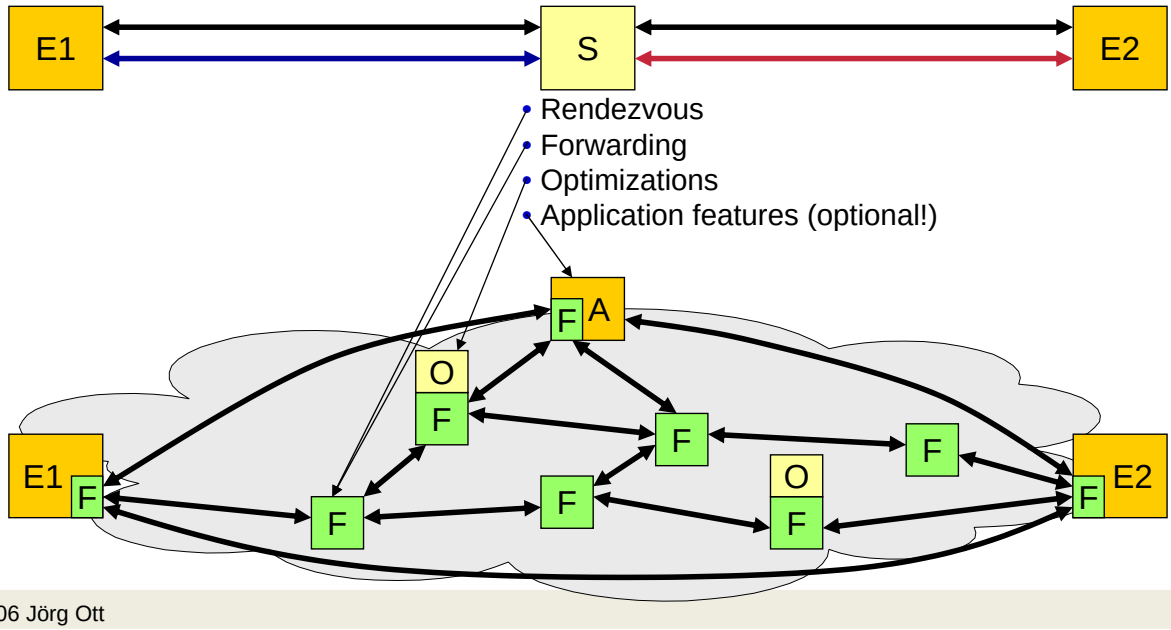
when designing protocols exclusively based upon asynchronous communications to deal with mobility

Assumption: Delay-tolerant Networking (DTN)

- Following the paradigm of asynchronous communications
  - Modeled after email
  - No instant end-to-end path necessary
  - Support for unicasting and multicasting
- Payload units ("bundles") of virtually arbitrary size
  - Forwarded using hop-by-hop reliability between DTN nodes
- Store-and-(carry-and-)forward
  - Workable with infrastructure routers, ad-hoc among endpoints, or both
  - DTN nodes may take immediate action on a bundle or delay forwarding
  - Routing decisions based upon known (present) or potential (future) paths
  - Forwarding of one or more copies of the unit when paths become available
    - Deterministic or probabilistic (e.g., epidemic) routing
- No need for continuous e2e paths

Endpoints and Intermediaries

- Endpoints: support symmetric operation
- Intermediaries: focus on common functions
  - Specific support should be opportunistic, but not create dependencies



Protocol Operation

- Explicit end-to-end state maintenance
  - Independence of lower layers
- Protocol operation
  - Self-contained messages
    - Note that MTU size is no longer an issue with DTN
  - Explicit context indication for operations (or idempotent ones)
    - Allow filtering duplicates and outdated ones
  - Persistent identifiers: e.g., URIs
  - Avoid interactive infrastructure use: e.g., late binding
  - Explicit support for intermediaries
- Some security considerations
  - Separate message contents (“substance”) from transaction parts
  - Messages need to be self-protecting (e.g., S/MIME)
  - DoS: Limit resource utilization per message
  - Open issue: key distribution, validation, and revocation

Soft Factors

- Look and Feel
  - Maintain the same basic look and feel as established applications when moving towards asynchronous communication
  - Yet provide better cues on progress and likely temporary error conditions
  - Introduce moderate controls
- Performance
  - When well connected, asynchronous application protocols should perform similarly to synchronous ones
    - User won’t (be able to) switch “mode of operation” between “mobile” and “fixed”
- Acceptance barriers
  - Built-in migration path: requires gateways (= intermediaries)
  - Perceived gain already for early adopters
  - Ad-hoc networking: users must be willing (motivated) to cooperate

Conclusion

- Supporting mobility means dealing with disconnections, non-existing end-to-end paths, and the resulting delays
  - Essentially: robustness in the presence of (temporary) failures
- Many applications' semantics are delay-tolerant, protocols aren't
  - Redesign protocols towards asynchronous e2e operation
  - Self-contained and context-explicit operations
  - Explicit support for intermediaries
  - Reduce dependency on specific intermediaries (optimization only)
- DTN concepts may offer generic intermediary functionality
  - Nice experimentation platform providing many useful features
  - Different paradigm requires fundamentally rethinking protocol design