15-744: Computer Networking

L-20 Data-Oriented Networking

Outline

• Data-oriented Networking
• DTNs

Data-Oriented Networking Overview

• In the beginning...
  – First applications strictly focused on host-to-host interprocess communication:
    • Remote login, file transfer, ...
  – Internet was built around this host-to-host model.
  – Architecture is well-suited for communication between pairs of stationary hosts.
• ... while today
  – Vast majority of Internet usage is data retrieval and service access.
  – Users care about the content and are oblivious to location.
    They are often oblivious as to delivery time:
    • Fetching headlines from CNN, videos from YouTube, TV from Tivo
    • Accessing a bank account at “www.bank.com”.

To the beginning...

• What if you could re-architect the way “bulk” data transfer applications worked
  • HTTP
  • FTP
  • Email
  • etc.
• ... knowing what we know now?
Innovation in Data Transfer is Hard

- Imagine: You have a novel data transfer technique
- How do you deploy?
  - Update HTTP. Talk to IETF. Modify Apache, IIS, Firefox, Netscape, Opera, IE, Lynx, Wget, ...
  - Update SMTP. Talk to IETF. Modify Sendmail, Postfix, Outlook…
  - Give up in frustration

Data-Oriented Network Design

New Approach: Adding to the Protocol Stack

Data Transfer Service

- Transfer Service responsible for finding/transferring data
  - Transfer Service is shared by applications
  - How are users, hosts, services, and data named?
  - How is data secured and delivered reliably?
  - How are legacy systems incorporated?
**Naming Data (DOT)**

- Application defined names are not portable
- Use content-naming for globally unique names
- Objects represented by an OID
  
  ![Diagram](#)
  
  **Cryptographic Hash**
  
  File → OID

- Objects are further sub-divided into “chunks”

  ![Diagram](#)

- Secure and scalable!

**Naming Data (DONA)**

- All objects are named based only on their data
- Objects are divided into chunks based only on their data

  ![Diagram](#)

- Object “A” is named the same
  - Regardless of who sends it
  - Regardless of what application deals with it

- Similar parts of different objects likely to be named the same
  - e.g., PPT slides v1, PPT slides v1 + extra slides
  - First chunks of these objects are same

**Similar Files: Rabin Fingerprinting**

- File Data
  
  ![Diagram](#)

- Rabin Fingerprints
  
  Hash 1
  
  Given Value
  
  Natural Boundary

- Natural Boundary

**Naming Data (DONA)**

- Names organized around principals.
- Names are of the form P : L.
  - P is cryptographic hash of principal’s public key, and
  - L is a unique label chosen by the principal.
- Granularity of naming left up to principals.
- Names are “flat”.

---

3
Self-certifying Names

• A piece of data comes with a public key and a signature.
• Client can verify the data did come from the principal by
  • Checking the public key hashes into P, and
  • Validating that the signature corresponds to the public key.
• Challenge is to resolve the flat names into a location.

Name Resolution (DONA)

• Resolution infrastructure consists of Resolution Handlers.
  • Each domain will have one logical RH.
• Two primitives FIND(P:L) and REGISTER (P:L).
  • FIND(P:L) locates the object named P:L.
  • REGISTER messages set up the state necessary for the RHs to route FINDs effectively.

Locating Data (DOT)

Locating Data (DONA)

REGISTER state
★ FIND being routed
Establishing REGISTER state

- Any machine authorized to serve a datum or service with name P:L sends a REGISTER(P:L) to its first-hop RH

- RHs maintain a registration table that maps a name to both next-hop RH and distance (in some metric)

- REGISTERs are forwarded according to interdomain policies.
  - REGISTERs from customers to both peers and providers.
  - REGISTERs from peers optionally to providers/peers.

Forwarding FIND(P:L)

- When FIND(P:L) arrives to a RH:
  - If there’s an entry in the registration table, the FIND is sent to the next-hop RH.
  - If there’s no entry, the RH forwards the FIND towards its provider.
  - In case of multiple equal choices, the RH uses its local policy to choose among them.

Interoperability: New Tradeoffs

Interoperability: Datagrams vs. Data Blocks
Outline

• Data-oriented Networking

• DTNs

Unstated Internet Assumptions

• Some path exists between endpoints
  • Routing finds (single) “best” existing route

• E2E RTT is not very large
  • Max of few seconds
  • Window-based flow/cong ctl. work well

• E2E reliability works well
  • Requires low loss rates

• Packets are the right abstraction
  • Routers don’t modify packets much
  • Basic IP processing

New Challenges

• Very large E2E delay
  • Propagation delay = seconds to minutes
  • Disconnected situations can make delay worse

• Intermittent and scheduled links
  • Disconnection may not be due to failure (e.g. LEO satellite)
  • Retransmission may be expensive

• Many specialized networks won’t/can’t run IP

IP Not Always a Good Fit

• Networks with very small frames, that are connection-oriented, or have very poor reliability do not match IP very well
  • Sensor nets, ATM, ISDN, wireless, etc

• IP Basic header – 20 bytes
  • Bigger with IPv6

• Fragmentation function:
  • Round to nearest 8 byte boundary
  • Whole datagram lost if any fragment lost
  • Fragments time-out if not delivered (sort of) quickly
IP Routing May Not Work

- End-to-end path may not exist
  - Lack of many redundant links [there are exceptions]
  - Path may not be discoverable [e.g. fast oscillations]
  - Traditional routing assumes at least one path exists, fails otherwise
- Insufficient resources
  - Routing table size in sensor networks
  - Topology discovery dominates capacity
- Routing algorithm solves wrong problem
  - Wireless broadcast media is not an edge in a graph
  - Objective function does not match requirements
    - Different traffic types wish to optimize different criteria
    - Physical properties may be relevant (e.g. power)

What about TCP?

- Reliable in-order delivery streams
- Delay sensitive [6 timers]:
  - connection establishment, retransmit, persist, delayed-ACK, FIN-WAIT, (keep-alive)
- Three control loops:
  - Flow and congestion control, loss recovery
- Requires duplex-capable environment
  - Connection establishment and tear-down

Performance Enhancing Proxies

- Perhaps the bad links can be ‘patched up’
  - If so, then TCP/IP might run ok
  - Use a specialized middle-box (PEP)
- Types of PEPs [RFC3135]
  - Layers: mostly transport or application
  - Distribution
  - Symmetry
  - Transparency

TCP PEPs

- Modify the ACK stream
  - Smooth/pace ACKS → avoids TCP bursts
  - Drop ACKs → avoids congesting return channel
  - Local ACKs → go faster, goodbye e2e reliability
  - Local retransmission (snoop)
  - Fabricate zero-window during short-term disruption
- Manipulate the data stream
  - Compression, tunneling, prioritization
Architecture Implications of PEPs

- End-to-end “ness”
  - Many PEPs move the ‘final decision’ to the PEP rather than the endpoint
  - May break e2e argument [may be ok]
- Security
  - Tunneling may render PEP useless
  - Can give PEP your key, but do you really want to?
- Fate Sharing
  - Now the PEP is a critical component
  - Failure diagnostics are difficult to interpret

Routing asymmetry
- Stateful PEPs generally require symmetry
- Spacers and ACK killers don’t

Mobility
- Correctness depends on type of state
- (similar to routing asymmetry issue)

Delay-Tolerant Networking Architecture

- Goals
  - Support interoperability across ‘radically heterogeneous’ networks
  - Tolerate delay and disruption
    - Acceptable performance in high loss/delay/error/disconnected environments
    - Decent performance for low loss/delay/errors
- Components
  - Flexible naming scheme
  - Message abstraction and API
  - Extensible Store-and-Forward Overlay Routing
  - Per-(overlay)-hop reliability and authentication

Disruption Tolerant Networks
Disruption Tolerant Networks

Naming Data (DTN)

- Endpoint IDs are processed as names
  - refer to one or more DTN nodes
  - expressed as Internet URI, matched as strings
- URIs
  - Internet standard naming scheme [RFC3986]
  - Format: `<scheme> : <SSP>`
- SSP can be arbitrary, based on (various) schemes
- More flexible than DOT/DONA design but less secure/scalable

Naming

- Support ‘radical heterogeneity’ using URI’s:
  - `{scheme ID (allocated), scheme-specific-part}
  - associative or location-based names/addresses optional
  - Variable-length, can accommodate “any” net’s names/addresses
- Endpoint IDs:
  - multicast, anycast, unicast
- Late binding of EID permits naming flexibility:
  - EID “looked up” only when necessary during delivery
  - contrast with Internet lookup-before-use DNS/IP

Message Abstraction

- Network protocol data unit: bundles
  - “postal-like” message delivery
  - coarse-grained CoS [4 classes]
  - origination and useful life time [assumes sync’d clocks]
  - source, destination, and respond-to EIDs
- Options: return receipt, “traceroute”-like function, alternative reply-to field, custody transfer
- fragmentation capability
- overlay atop TCP/IP or other (link) layers [layer ‘agnostic’]
- Applications send/receive messages
  - “Application data units” (ADUs) of possibly-large size
  - Adaptation to underlying protocols via ‘convergence layer’
  - API includes persistent registrations
DTN Routing

- DTN Routers form an overlay network
  - only selected/configured nodes participate
  - nodes have persistent storage

- DTN routing topology is a **time-varying** multigraph
  - Links come and go, sometimes predictably
  - Use any/all links that can possibly help (multi)
  - Scheduled, Predicted, or Unscheduled Links
    - May be direction specific [e.g. ISP dialup]
    - May learn from history to predict schedule

- Messages fragmented based on dynamics
  - Proactive fragmentation: optimize contact volume
  - Reactive fragmentation: resume where you failed

Example Routing Problem

Example Graph Abstraction

The DTN Routing Problem

- **Inputs**: topology (multi)graph, vertex buffer limits, contact set, message demand matrix (w/priorities)

- An **edge** is a possible opportunity to communicate:
  - One-way: \((S, D, c(t), d(t))\)
  - \((S, D)\): source/destination ordered pair of contact
  - \(c(t)\): capacity (rate); \(d(t)\): delay
  - A **Contact** is when \(c(t) > 0\) for some period \([i_k, i_{k+1}]\)

- Vertices have buffer limits; edges in graph if ever in any contact, multigraph for multiple physical connections

- **Problem**: optimize some metric of delivery on this structure
  - Sub-questions: what metric to optimize?, efficiency?
Routing Solutions - Replication

- “Intelligently” distribute identical data copies to contacts to increase chances of delivery
  - Flooding (unlimited contacts)
  - Heuristics: random forwarding, history-based forwarding, predication-based forwarding, etc. (limited contacts)
- Given “replication budget”, this is difficult
  - Using simple replication, only finite number of copies in the network [Juang02, Grossglauser02, Jain04, Chaintreau05]
  - Routing performance (delivery rate, latency, etc.) heavily dependent on “deliverability” of these contacts (or predictability of heuristics)
  - No single heuristic works for all scenarios!

Using Erasure Codes

- Rather than seeking particular “good” contacts, “split” messages and distribute to more contacts to increase chance of delivery
  - Same number of bytes flowing in the network, now in the form of coded blocks
  - Partial data arrival can be used to reconstruct the original message
    - Given a replication factor of $r$, in theory any $1/r$ code blocks received can be used to reconstruct original data
  - Potentially leverage more contacts opportunity that result in lowest worse-case latency
- Intuition:
  - Reduces “risk” due to outlier bad contacts
Erasure Codes

- Message n blocks
  - Encoding
  - Opportunistic Forwarding
  - Decoding
  - Message n blocks

DTN Security

- Payload Security Header (PSH) end-to-end security header
- Bundle Authentication Header (BAH) hop-by-hop security header

So, is this just e-mail?

<table>
<thead>
<tr>
<th></th>
<th>routing</th>
<th>flow</th>
<th>multi-app</th>
<th>security</th>
<th>reliable</th>
<th>priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>e-mail</td>
<td>N (static)</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>DTN</td>
<td>Y</td>
<td>Y (static)</td>
<td>Y</td>
<td>Y</td>
<td>opt</td>
<td>opt</td>
</tr>
</tbody>
</table>
Performance: Opportunistically Aggregate Upstream Bandwidth

- Challenges:
  - Multi-hop: decide whether to send upstream or forward one extra hop
  - Requires both data-oriented design and opportunistic forwarding

Sender broadcasts to available neighbors

Neighbors decide whether to ship the pieces upstream