Delay-Tolerant Networking: Architecture and Applications

Kevin Fall

Intel Research, Berkeley

kfall@intel-research.net

Stanford Networking Seminar 4-DEC-2003

Intel Research Berkeley

Outline

- Delay Tolerant Network Architecture
  – *Why the Internet Architecture is not a 'one-size-fits-all' solution*
- Some things to do with your DTN…
  – Networking for developing regions of the world

    (a new NSF-sponsored project)

Unstated Internet Assumptions

- End-to-end RTT is not terribly large
  - A few seconds at the very most [typ < 500ms]
  - (window-based flow/congestion control works)
- Some path exists between endpoints
  - Routing finds single "best" existing route
    - [ECMP is an exception]
- E2E Reliability using ARQ works well
  - True for low loss rates (under 2% or so)
- Packet switching is the right abstraction
  - Internet/IP makes packet switching interoperable

Non-Internet-Like Networks

- Stochastic mobility
  - Military/tactical networks
  - Mobile routers w/disconnection (e.g. ZebraNet)
- Periodic/predictable mobility
  - Spacecraft communications
  - Busses, mail trucks, police cars, etc. (InfoStations)
- "Exotic" links
  - Deep space [40+ min RTT; episodic connectivity]
  - Underwater [acoustics: low capacity, high error rates & latencies]

New challenges…

- Very Large Delays
  - Natural prop delay could be seconds to minutes
  - If disconnected, may be (effectively) much longer
- Intermittent/Scheduled/Opportunistic Links
  - Scheduled transfers can save power and help congestion; scheduling common for esoteric links
- High Link Error Rates / Low Capacity
  - RF noise, light or acoustic interference, LPI/LPD concerns
- Different Network Architectures
  - Many specialized networks won't/can't ever run IP

What to Do?

- Some problems surmountable using Internet/IP
  - ‘cover up’ the link problems using PEPs
  - Mostly used at “edges,” not so much for transit
- Performance Enhancing Proxies (PEPs):
  - Do “something” in the data stream causing endpoint (TCP/IP) systems to not notice there are problems
  - Lots of issues with transparency– security, operation with asymmetric routing, etc.
- Some environments *never* have an e2e path
  - Consider an approach tolerating disconnection, etc...

Delay-Tolerant Networking Architecture

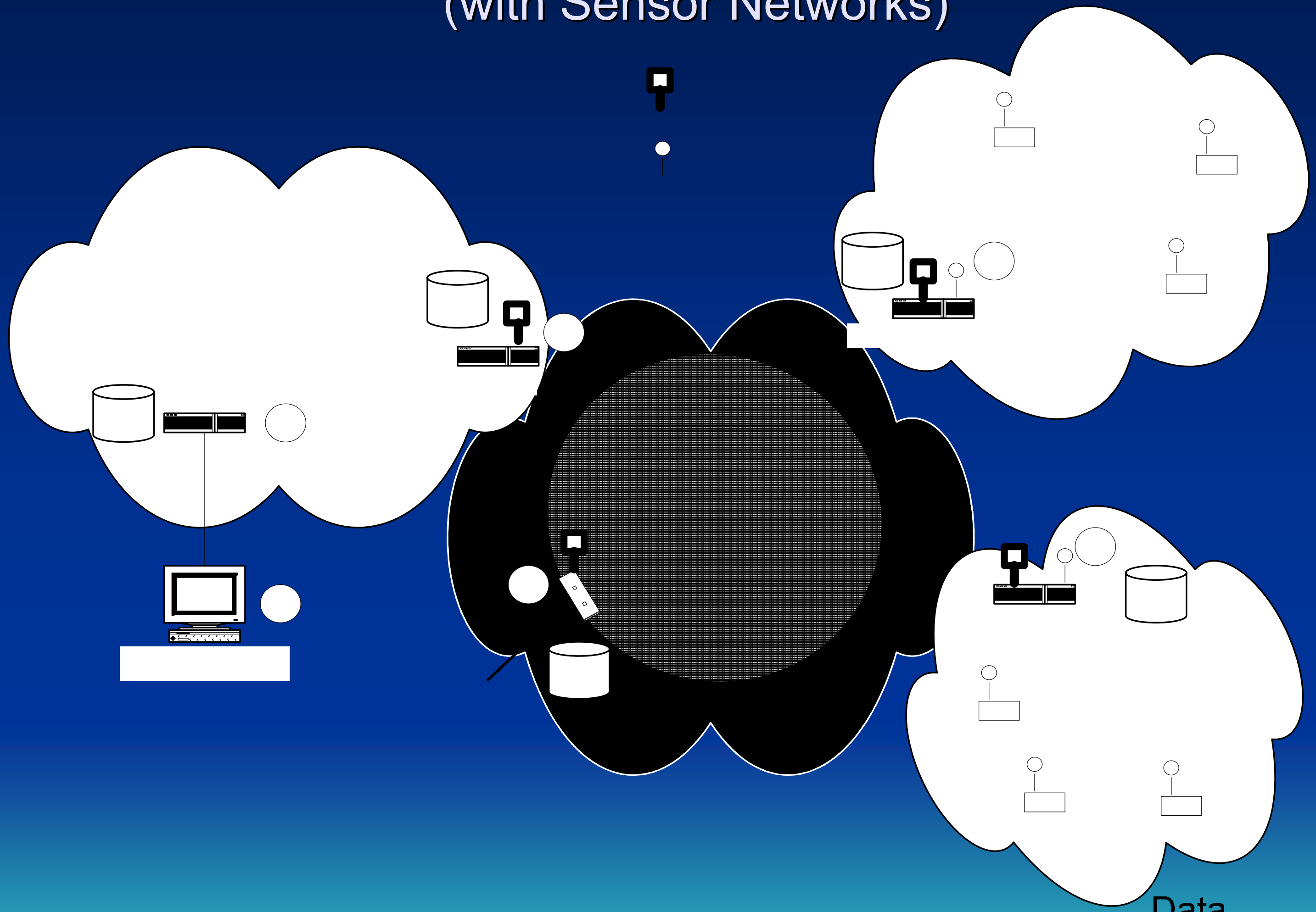
- Goals
  - Support interoperability across ‘radically heterogeneous’ networks
  - Acceptable performance in high loss/delay/error/disconnected environments
  - Decent performance for low loss/delay/errors
- Components
  - Flexible naming scheme with *late binding*
  - Message overlay abstraction and API
  - Routing and link/contact scheduling w/CoS
  - Per-(overlay)-hop reliability and authentication

Naming and Regions

- Support heterogeneity using *regions*:
  - Instance of an internet, not so radical inside a region
  - Common naming and protocol conventions
- Endpoint Name: ordered pair **`{R,L}`**
  - **`R`**: routing region name [globally valid]
  - **`L`**: region-specific ID, opaque outside region **`R`**
- ***Late binding*** of **`L`** permits naming flexibility:
  - Associative or location-oriented names [URN vs URL]
    - Internet-style URI gives both [see RFC2396]
  - May encompass esoteric routing [e.g. diffusion]
- *Issue*: make **`R,L`** compressible in transit networks

Example Regions

(with Sensor Networks)



Intel Research Berkeley

Message Overlay Abstraction

- End-to-End Message Service: “Bundles”
  - “postal-like” message delivery over regional transports with coarse-grained CoS [4 classes]
  - *Options*: return receipt, “traceroute”-like function, alternative reply-to field, custody transfer
  - Supportable on nearly any type of network
- Applications send/receive <u>bundles</u>
  - “Application data units” of possibly-large size
  - May require framing above some transport protocols
  - API supports response processing long after request was sent (application *re-animation*)

So, is this all just e-mail?

	naming/	routing	flow	multi-	security	reliable	priority
	late binding		contrl	app		delivery	
e-mail	Y	N	Y	N	opt	Y	N(Y)
DTN	Y	Y	Y	Y	opt	opt	Y

- Many similarities to (abstract) e-mail service
- Primary difference involves routing and API
- E-mail depends on an underlying layer's routing:
  - Cannot generally move messages closer to their destinations in a partitioned network
  - In the Internet (SMTP) case, not disconnection-tolerant or efficient for long RTTs due to "chattiness"
- E-mail security authenticates only user-to-user

Routing on Dynamic Graphs

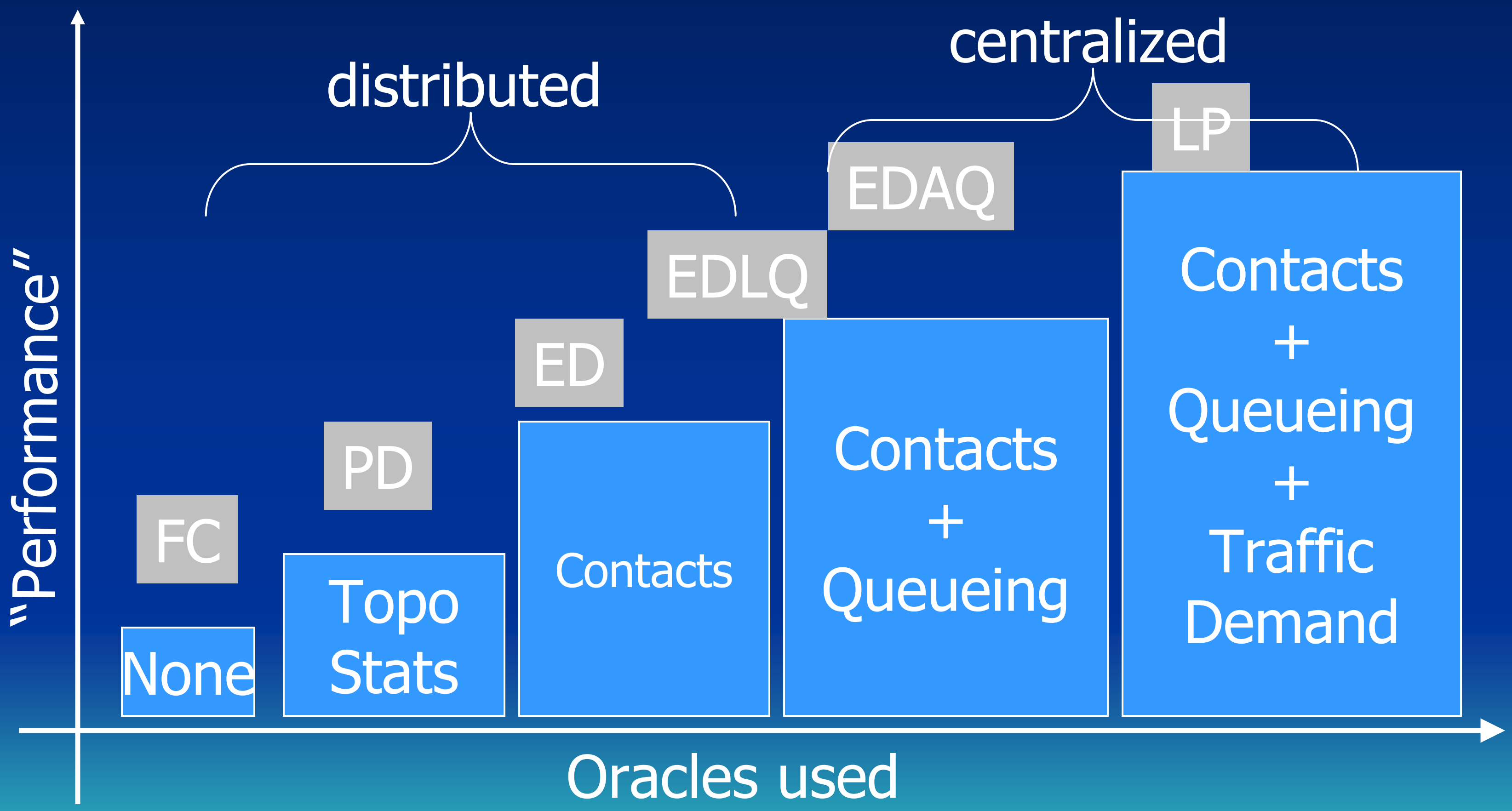
- DTN routing takes place on a time-varying topology
  - Links come and go, sometimes predictably
- Scheduled and Unscheduled Links
  - May be direction specific [e.g. ISP dialup]
  - May learn from history to predict schedule
- Link ``*Predictability continuum*''
  - S/U represents extreme cases regarding the expected availability of a route to a destination
  - An intermediate "predicted" category may evolve as a result of statistical estimation
  - Represent by a entropy-like measure (?)

The DTN Routing Problem

- *Inputs*: topology (multi)graph, vertex buffer limits, contact set, message demand matrix (w/priorities)
- A ***contact*** is an opportunity to communicate:
  - One-way: (t_s, t_e, S, D, C, D)
  - (t_s, t_e): contact start and end times
  - (S, D): source/destination ordered pair of contact
  - C: capacity (rate; assume const over (t_s, t_e)); D: delay
- 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-question: what metric to optimize?

Routing Algorithm Study

Sushant Jain (UW)



Custody Transfer

- Bundle routers use persistent storage
  - May provide *custody transfer service* if so requested
  - If so, will try “very hard” to not discard messages for which it has accepted custody
  - Accepting custody for a bundle may involve a significant allocation of resources at a bundle router
- This raises some important questions:
  - What does flow and congestion control look like in one of these environments?
  - When should a bundle router avoid taking custody?
  - Given the hop-by-hop nature, if congestion control is figured out, does this also solve flow control?

Intel Research Berkeley

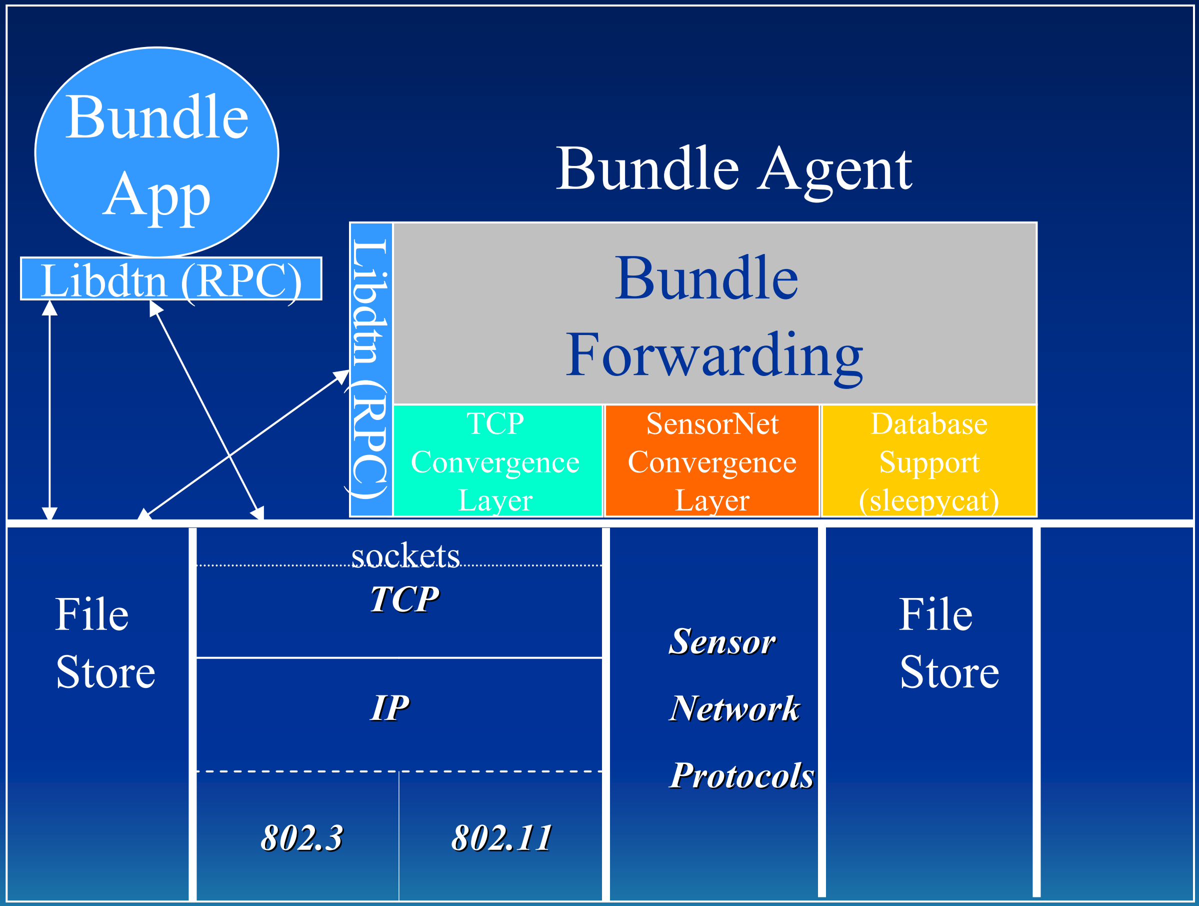
Flow and Congestion Control

- Control at coarse time scales (“filesystem full”)
  - Very high delay → pre-schedule/admission control
  - Small delay → dynamic flow control possible
  - Where does ‘traffic engineering’ end and ‘dynamic flow (congestion) control’ begin?
- In low-delay cases, which layer exerts FC?
  - Region-specific transports may support their own FC
  - Flow-control is logically hop-by-hop, so problem is to convert bundle-layer flow control to protocol-specific FC mechanism, which depends on transport
  - Multiplexing multiple bundles on one transport causes problems due to head-of-line-blocking like phenomena

DTN API

- RPC-based API is “split-phase” (libdtn)
  - RPC base allows for remote (dumb) clients
    - Apps are both clients and servers to RPC
  - sends decoupled from async receives
    - Request/response time may exceed longer than end-node lifetime
    - “Re-animation” capability to requestor or other
- Forwarder performs heavy lifting (bundledaemon)
  - Application (de)registrations
  - Executes convergence layers for send/receive
  - Bundle database maintenance
  - Basic routing functions

Forwarder Implementation



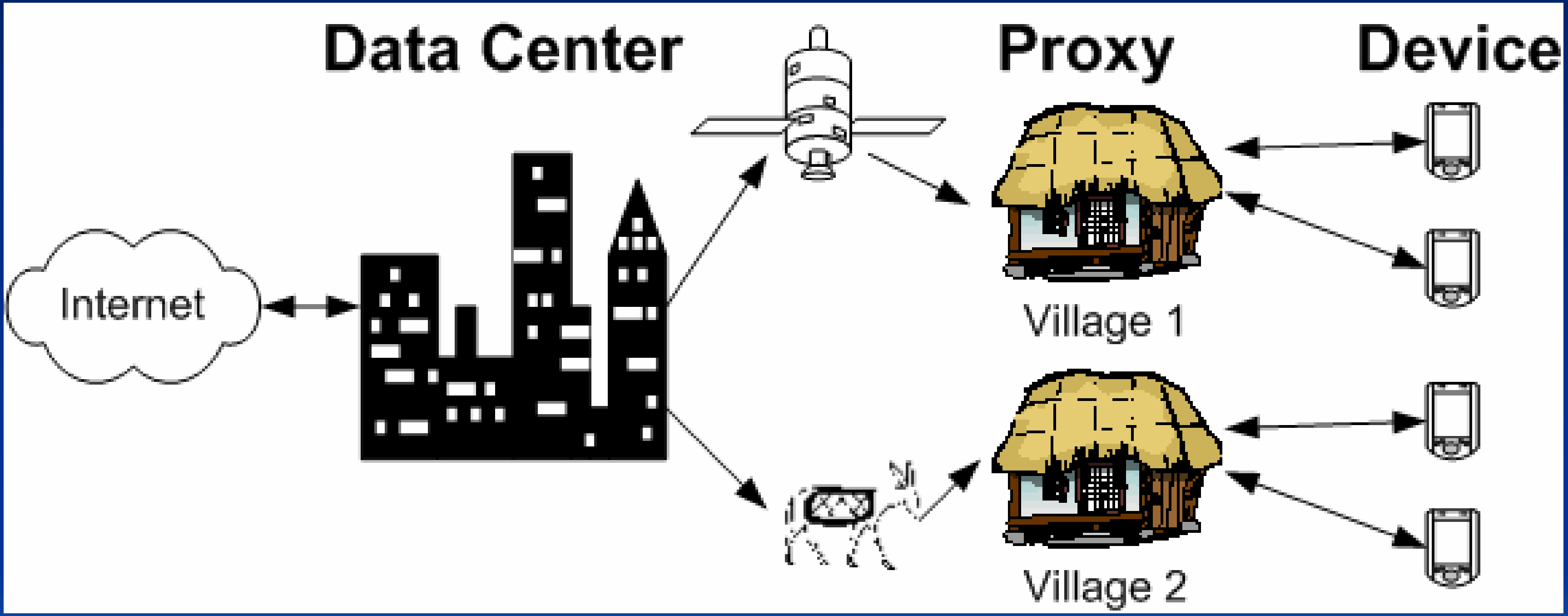
Intel Research

On to an application…

ICT for Billions (ICT4B)

- Information and Communication Technologies for Developing Regions of the World
- Networking focus: *intermittent networking*
  - *Mission-specific architecture and API*
  - *Multiple layers of network intermittency*

TIER "tiered" architecture



DTN and TIER

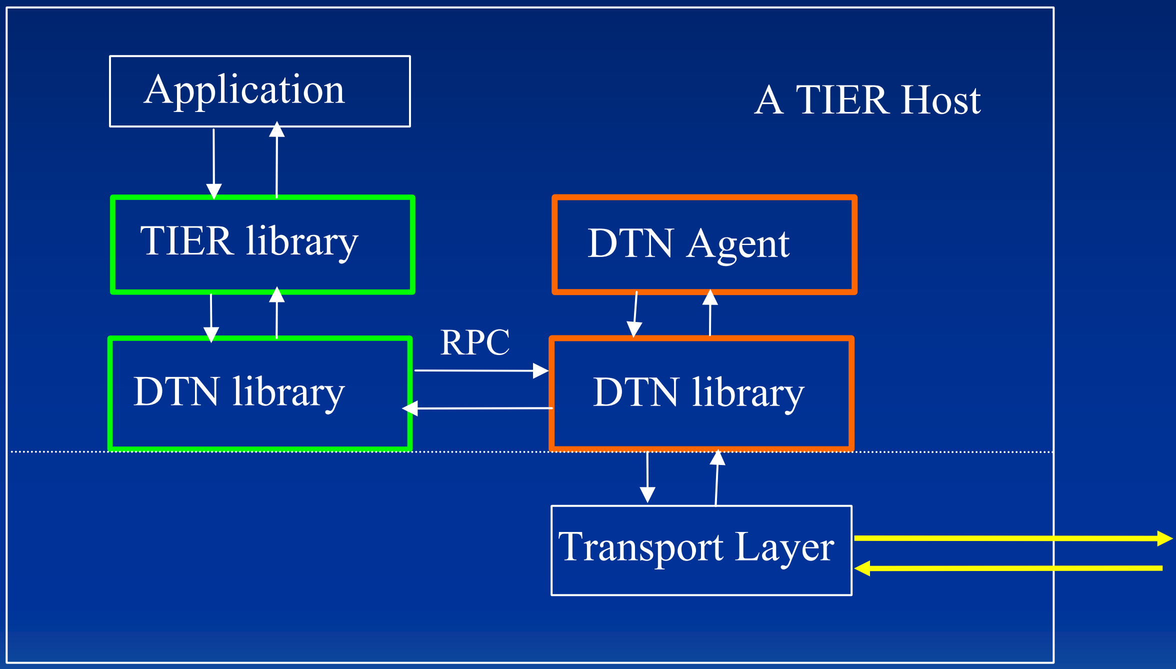
- DTN
  - Architecture and reference implementation of DTN
  - Further development supported by DARPA/ATO for military applications
- TIER (**T**echnology and **I**nfrastructure for **E**merging **R**egions) building on DTN
  - Specialized API for 3-tier architecture
  - E-mail type driver application

TIER API

- Asynchronous delivery of messages
- Multiple Traffic Classes
  - Events, Periodic messages, Two-way channel
- Use of preset configuration state variables
  - for simplifying specification of many different parameters
- Ability to get connectivity status from network
  - events: connection established, connection broken
- Discovery of network and proxies
- Generic caching infrastructure

Implementation Structure

- Use DTN agent for message transfer
  [www.dtnrg.org]
- Message = bundle
- Callbacks for:
  - received messages
  - connectivity changes



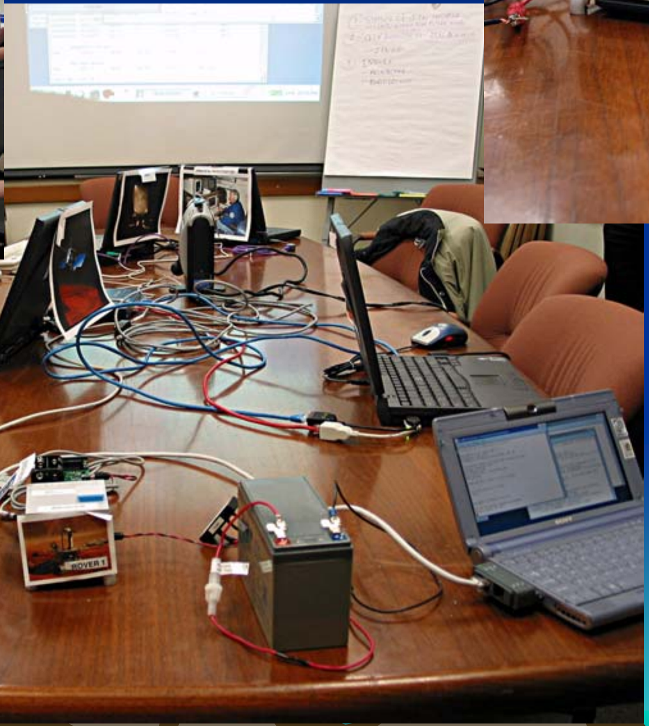
Mail4b Project Goals

- Evaluate DTN and Tier API and drive development direction
  - Create a realistic application
- Evaluate "tiered architecture"
  - Data Center, Proxies, Devices
  - Enable sharing of high-cost assets
  - Allow end-devices to be simple, low cost, and easy to use {maybe disposable}

Experiences to Date

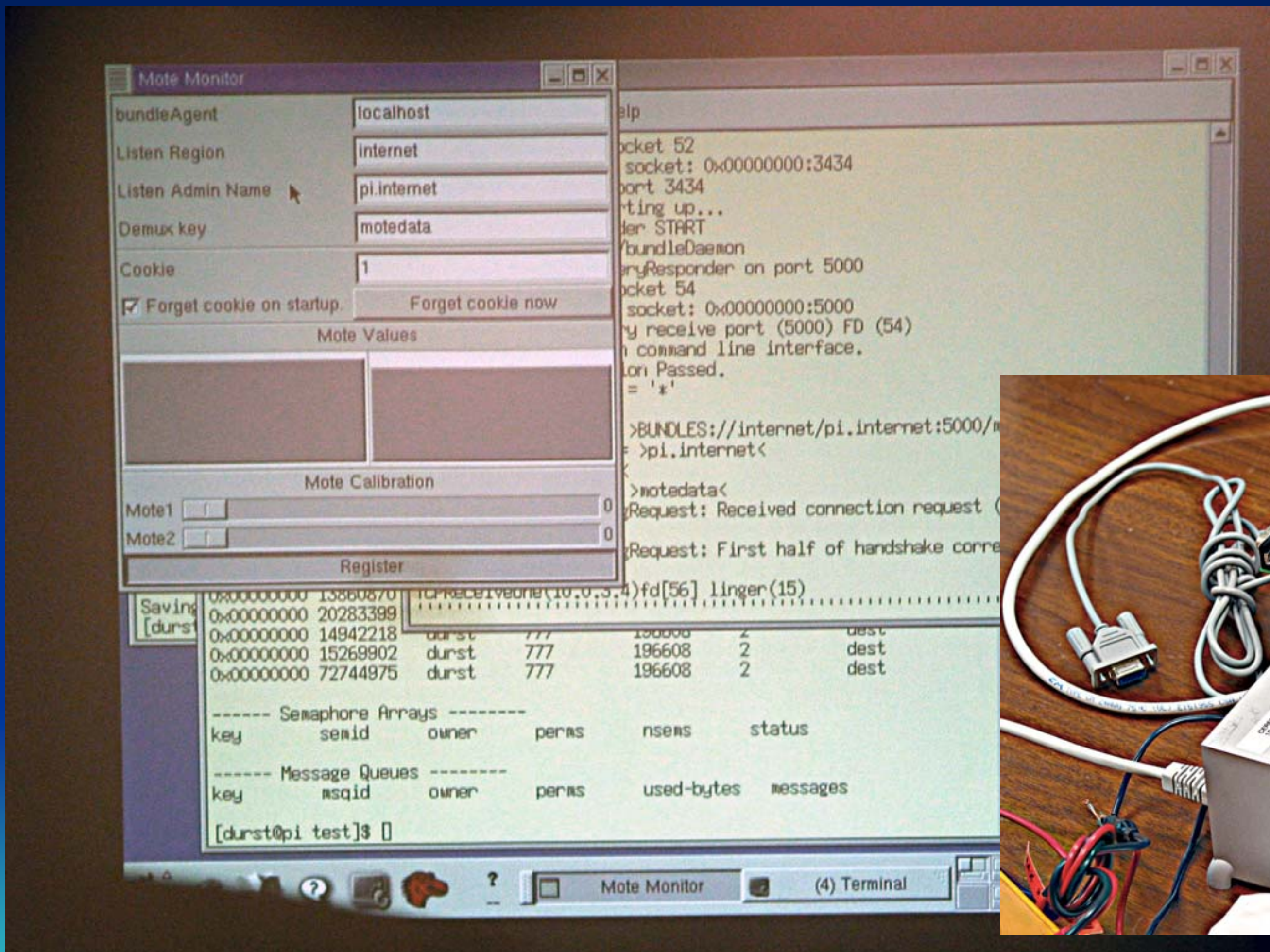
- First Mail4B implementation exposed key issues with infrastructure
  - Caching should be a basic feature of the TIER API
  - DTN should expose the connectivity state
    - device wants in-village and disconnected modes of operation
  - TIER API should have a “periodic” traffic class for status-style messages
- Future Work
  - More API extensions, Data Center clustering, deployment on PDA-class devices

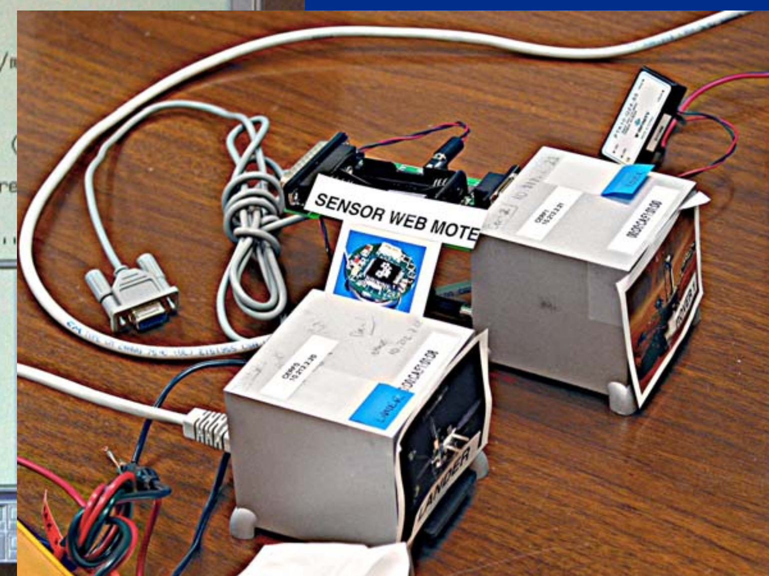
Demo (1)



Intel Research Berkeley

Demo (2)





Intel Research Berkeley

People and Places

- DTN Effort [DARPA/ATO, JPL, MITRE, MCI, Intel]
  - J. Alonso (SICS)
  - S. Burleigh, A. Hooke (NASA/JPL)
  - V. Cerf (MCI)
  - B. Durst, K. Scott (MITRE)
  - S. Jain (Univ of Washington)
- ICT/TIER Effort [NSF, UCB, ICSI, Intel]
  - E. Brewer, R. Patra, S. Nadevschi, M. Demmers, B. Du (UCB)
  - Anind Dey, K. Fall (IRB)

Intel Research Berkeley

http://www.dtnrg.org

Thank You!

http://tier.cs.berkeley.edu

Intel Research Berkeley

Backups

‘E-Mail4b’ Application

- Email as a “delay tolerant” application w/caching
  - Asynchronous by nature
  - Users don’t notice the delay if they only check mail intermittently
- Good fit for the TIER [Traffic Class] API
  - Mail naturally matches bulk data transfer
  - Other messages (e.g. status queries) match to small periodic message class

API Issues

- Configuration Parameters
  - Priority, reliability, timeout, rate, ACK reqd
- Traffic classes
  - General: Alert, data xfer, interactive
  - Special extra semantics: *squash*
    - drop older version if more recent message seen
- Naming of devices/proxies

Status

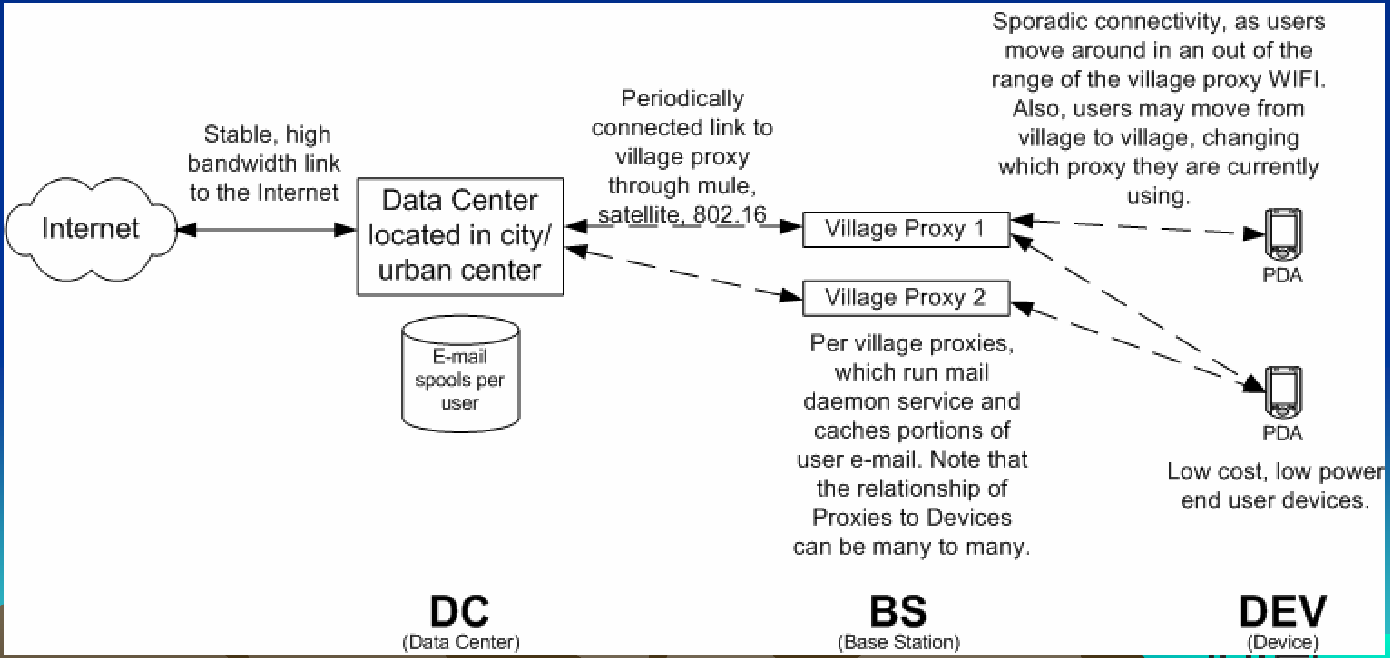
- IETF/IRTF DTNRG formed end of 2002
  - See http://www.dtnrg.org
- DTN Agent Source code released 3/2003
- SIGCOMM Paper presented 8/2003
- Several other documents (currently ID's):
  - DTNRG Architecture document
  - Bundle specification
  - Application of DTN in the IPN
- Basis for new DARPA DTN funding opportunity

Acknowledgements

- People (design & agent implementation):
  - Bob Durst, Keith Scott (MITRE)
  - Kevin Fall (Intel Research)
  - Sushant Jain (UW Intern), Rabin Patra (UCB Intern)
- More people (vision, design, commentary):
  - Vint Cerf (MCI)
  - Scott Burleigh, Adrian Hooke (NASA/JPL)
  - Juan Alonso (SICS)
  - Howard Weiss (SPARTA)
  - Forrest Warthman (Warththman)
  - Stephen Farrell (Ireland)
  - The *dtn-interest* mailing list

TIER Applications

- Email using TIER API
  - PDAs have sporadic connectivity
  - Proxies cache email from Data Centers for PDAs



Research challenges…

- Network Interface to Applications
  – Probably asynchronous
  – May be useful to label traffic class
- Scheduling/Routing for Disconnected Nets
  – Scheduled transfers can save power and help congestion; may have hybrid high/low delay systems
- Network Architecture Heterogeneity
  – How to make `radically heterogeneous' networks interoperate
- Do all this on highly affordable devices…

For more Information

- Delay Tolerant Networking Research Group
  - **http://www.dtnrg.org**
- Intel Research
  - **http://www.intel-research.net**
- IRTF Web Page:
  - **http://www.irtf.org**

kfall@intel-research.net

Intel Research Berkeley

Some Security Issues

- Primary focus: *infrastructure protection*
  - Verify transit authorization at each overlay hop
  - Need some public-key facility for doing this
  - “Core” bundle routers must not be required to know every end-user set of credentials
    - Too big/slow; may be disconnected– difficult to look up
- Compromise for scalability
  - ACLs and user keys contained at firs-hop ‘edge’ routers
  - Edge routers authenticate and re-sign messages in their own keys
  - Next-hop routers need only check keys of its O(log n) [or maybe O(1)] neighbors

Security Issue Details

- Effect of a router compromise:
  - Router compromise could result in traffic being carried from that point onward
  - Router cannot completely masquerade as sender
    - Sending user still has its own private/public pair
- Compromise for scalability
  - ACLs and user keys contained at firs-hop ‘edge’ routers
  - Edge routers authenticate and re-sign messages in their own keys
  - Next-hop routers need only check keys of its O(log n) [or maybe O(1)] neighbors

Intel Research Berkeley

Authentication of Fragments

- Consider xfer of bundle Z along link A->B
  - Z was signed by sender, but is also signed by A for transit through B
  - A->B link goes unavailable, but much of Z made it
- How to authenticate on fragments
  - Is there a keyed hash function that can take a substring (prefix) of a message and still somehow verify the signature [without using the 'dice into chunks' model]?

Some Networks with Impairments

- LEO satellites [periodic connectivity]
- Sensor networks connected via "mules"
- Roaming underwater vehicles using acoustic modems
- Deep space communications [beyond near-Earth orbit]
- Some military ad hoc networks

DTN Architecture Drivers -- Assumptions

- No contemporaneous e2e path may ever exist between sender and receiver
- DTN Routers are equipped with significant persistent storage
- Retransmission may be *very* expensive
- Round-trip times could range from milliseconds to days
- Early prevention of unauthorized use of the network is desirable

DTN Architecture Drivers – Hard Problems

- *Reliability* **and** *congestion management* **in high-delay, high-error, and disconnected environments**
- *Path selection* **and** *scheduling* **in graphs with opportunistic and periodic contacts (time-varying directed edges)**
- *Interoperability* **across dissimilar protocol stacks**

Intel Research Berkeley

The Internet for all…….?

- Lots of projects to connect ‘Internet’ (the Web)
  – But not all applications require the Web
    - Web does not equal “The Internet”
    - (e.g. e-mail = most popular Internet application)
  – ‘Always on’ networking may be hard
    - High installation and operational costs
    - Poor connectivity reflected in poor application performance

- Assuming *network intermittency* may be better…

Routing in a DTN

- Scheduled (known) / Unscheduled (opportunistic)
  - S/U characterization may be direction-specific
  - Consider the two ends of a user/ISP link
- Formulation as an LP (ideal case):
  - Minimize the *evacuation* time
  - Constraints on time, buffers, messages, priority
  - Several non-ideal options under investigation
- Predictability continuum:
  - Intermediate "predicted" category may evolve as a result of statistical estimation
  - Represent by a entropy-like measure (?)

Implementation and API

- C/Java RPC-based API is “split-phase” (callbacks)
  - DTN agent need not be co-located with clients
  - Apps execute as RPC clients and servers
- Decoupled arrival and app delivery
  - Generalizes e-mail mailboxes
  - Can specify action on receipt [drop,hold,exec]
    - Apps are both clients and servers to RPC
- DTN agent performs heavy lifting
  - DB for app (de)registrations, bundle send/recv/demux
  - Name resolution in destination region as required
  - Basic routing and scheduling functions
  - Custody transfer
  - Authentication and access control (if requested)

Status

- DTN is an architecture for:
  - Internetworking in frequently-disconnected networks
  - Interconnecting 'radically heterogeneous' networks
- It evolved from the IPN Architecture
- There is a prototype implementation
  - ~20K lines of C code and some JAVA
  - Demonstrated as basis for query processing in disconnected sensor network
- There is an IRTF research group (DTNRG)

Acknowledgements

- People (implementers):
  - Bob Durst (MITRE)
  - Scott Burleigh (NASA/JPL)
  - Keith Scott (MITRE)
- More people (vision, design, commentary):
  - Vint Cerf (MCI)
  - Adrian Hooke (NASA/JPL)
  - Eric Travis (GST)
  - Howard Weiss (SPARTA)
  - The *dtn-interest* mailing list at IRB

For more Information

- Delay Tolerant Networking Research Group
  - http://www.dtnrg.org
- Internet Research Task Force
  - http://www.irtf.org
- DTN Mailing list
  - dtn-interest@mailman.dtnrg.org
- Interplanetary Internet SIG (ISOC group)
  - http://www.ipnsig.org

Network Intermittency

- *Intermittency* – the inability to establish or maintain a contemporaneous e2e association
  - Causes: inadequate infrastructure, power failure/scheduling, configuration errors
  - Expected to be especially important in 3rd world…
- Applications and networking layer should accommodate network intermittency
  - Planned or not

- Networking should be ***Delay Tolerant***

Conclusions

- 3-TIER Architecture
  – Data centers, Villages, Portables
- Networking should accommodate *network intermittency* between tiers
  – Expected to be cheaper and more common for our expected deployments
  – Building upon pre-existing work in Delay Tolerant Networking (DTNRG)
  – Enhancements: Discovery, caching, traffic class API, etc…

Naming Challenges

- Structure of **R** (region name)
  - Variable length, hierarchical, centrally? allocated
  - Could likely use DNS namespace w/out mechanism
- How does a sender know/learn destination's R?
  - "just does" (like well-known port)
  - Some centralized or distributed service
- What semantic rules really apply to **L**?
  - Associative and location-based names seem useful
    - Associative – "send to Kevin's pager" [who looks up?]
    - Location – "send to pager [addr: p103x] via Inet gw
- Associative naming requires mapping server
  - Unworkable in high-delay/disconn environment

Mail4b "tiered" architecture

- Data Center in major city
  - Permanent, reliable database of mail data, registrations, etc
  - Always-up connection to the internet, intermittent connections to each village
- Proxy in each village
  - Local cache for mail data
  - Wireless local-area networking to communicate with devices
- Device
  - Low-cost PDA class device with wireless network

TIER API Backgorund

- Three-tiered architecture for ICT4B
  - Data centers [reliable storage/comms]
  - Proxies [relays/cache w/persistent storage]
  - Handheld devices [unreliable storage/comms]
- Intermittent connectivity between tiers
  - 802.11 ad-hoc and p2p, LEO/GEO satellite connections, mules