Disruption/Delay-Tolerant Networking Tutorial

Kevin Fall & Michael Demmer
Intel Research and UC Berkeley

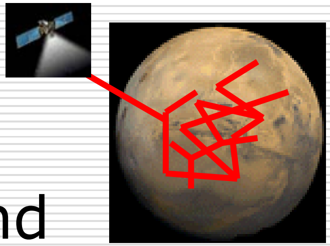
http://WWW.DTNRG.ORG

May 22, 2006 / Mobihoc 2006 / Florence, Italy

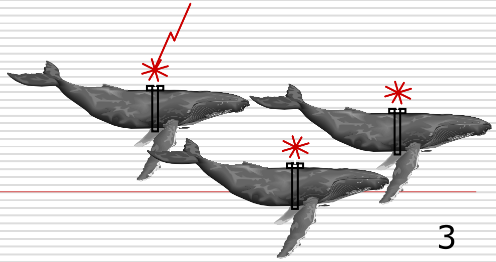
Outline

- ***Challenged Networks and the Internet Architecture***
- *DTN Architecture Overview*
- *DTN People & Projects*
- *DTN Research Summary*
- *DTN Reference Implementation*

What are Challenged Networks?



- Unusual
  - Containing features or requirements a networking architecture designer would find surprising or difficult to reason about
- Challenged
  - An operating environment making communications difficult
- *Examples*: mobile, power-limited, far-away nodes communicating over poorly or intermittently-available links



RFC1149 : A Challenged Internet

- "…encapsulation of IP datagrams in avian carriers" (i.e. birds, esp carrier pigeons)
- Delivery of datagram:
  - Printed on scroll of paper in hexadecimal
  - Paper affixed to AC by duct tape
  - On receipt, process is reversed, paper is scanned in via OCR

Implementation of RFC1149

CPIP: Carrier Pigeon Internet Protocol

- See http://www.blug.linux.no/rfc1149/

Ping Results

```
Script started on Sat Apr 28 11:24:09 2001
vegard@gyversalen:~$ /sbin/ifconfig tun0
tun0      Link encap:Point-to-Point Protocol
          inet addr:10.0.3.2  P-t-P:10.0.3.1  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:150  Metric:1
          RX packets:1 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          RX bytes:83 (83.0 b)  TX bytes:168 (168.0 b)

vegard@gyversalen:~$ ping [illegible] 10.0.3.1
PING 10.0.3.1 (10.0.3.1): 56 data bytes
64 bytes from 10.0.3.1: icmp_seq=0 ttl=255 time=6165731.1 ms
64 bytes from 10.0.3.1: icmp_seq=4 ttl=255 time=3211900.8 ms
64 bytes from 10.0.3.1: icmp_seq=2 ttl=255 time=5124922.8 ms
64 bytes from 10.0.3.1: icmp_seq=1 ttl=255 time=6388671.9 ms

--- 10.0.3.1 ping statistics ---
9 packets transmitted, 4 packets received, 55% packet loss
round-trip min/avg/max = 3211900.8/5222806.6/6388671.9 ms
vegard@gyversalen:~$ exit

Script done on Sat Apr 28 14:14:28 2001
```

Private Addresses

About 1.5 Hrs

High Loss

Internet Architecture

- Key design points
  - Packet abstraction is good
  - Fully-connected routing graph
  - Hierarchical address assignment
  - End-to-end reliability – dumb network
  - Management at the application layer
  - Security and accounting secondary (at ends)

Internet is a Packet Network

- Internet Protocol
  - Abstract IP datagram
    - Fragmentation function adapts this
  - Globally-unique IP addresses
    - Addresses are hierarchical to save routing table space
  - Store-and-forward
    - Short-term storage of a few packets
    - Drop on overload (typically “drop tail”)

Internet is Fully-Connected

- Internet Protocol
  - Routing
    - Implemented as an application on the Internet
    - Finds “best” (single) path among network prefixes
      - There should be lots of paths available, so pick one
    - No (transport-layer or higher) state in routers
  - Drop on failure
    - “No route to host” – failure of the abstraction due to failure of the environmental assumptions

Hierarchical Addresses

- Internet Protocol
  - Addresses
    - every interface has a 32-bit unique address
    - share a prefix with other nearby machines
      - subnets
      - CIDR and aggregation
  - Consequences
    - too few addresses –> IPv6 and NAT
    - mobility -> indirection

Reliability is End-to-End

- Fate sharing
  - If one endpoint dies, the other might as well too
    - Consistent with connections
    - Simple network infrastructure, sophisticated end hosts
    - End hosts should behave
- Re-transmission is an appropriate method to combat loss

Management at Application Layer

- Control is in-band
  - Subject to same anomalies as regular data
  - Subject to attacks
- Management capabilities depend on which apps are installed
  - A limited *de-facto* standard set
- Management is the last thing to be enabled

Security and Accounting

- Security as an add-on
  - Identity is not secured
  - Not implemented at a consistent layer
  - Traffic management (filtering) vs end-to-end authentication
    - Filtering limited/fragile, authentication may be burdensome
    - Middlebox problems
- Accounting
  - Difficult to account for and pay for use

Internet Assumptions

- E2E path doesn't have *really* long delay
  - Reacting to flow control in ½-RTT effective
  - Reacting to congestion in 1-RTT effective
- E2E path doesn't have *really* big, small, or asymmetric bandwidth
- Re-ordering might happen, but not much
- End stations don't cheat
- Links not very lossy (< 1%)
- Connectivity exists through *some* path
  - even MANET routing usually assumes this

More Internet Assumptions

- We are all among friends here
  - ‘security’ evolution from addresses to crypto
  - mostly an add-on [ok for transport; not for IP layer]
- Nodes don’t move around or change addresses
  - easy to assign addresses in hierarchy
  - thought to be important for scalability
- In-network storage is limited
  - not appropriate to store things long-term in network
- End-to-end principle
  - routers are ‘flakier’ than end hosts

Non-Internet-Like Networks

- Random and predictable node mobility
  - Military/tactical networks (clusters meet clusters)
  - Mobile routers w/disconnection (e.g. ZebraNet)
- Big delays, low bandwidth (high cost)
  - satellites (GEO, LEO / polar)
  - exotic links (deep space comms, underwater acoustics)
- Big delays, high bandwidth
  - Busses, mail trucks, delivery trucks, etc.

Challenged Networks…

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

Internet for Challenged Networks?

- What happens when one or more of the Internet assumptions don’t hold (strongly)?
  - Applications break / communication disabled
  - Applications have intolerable performance
  - System is not secure
- Let’s be more specific…

Comms System Challenges

- Loss-prone links
- Opportunistic and scheduled Links
- Links with large and/or variable delays
- Limited node uptime (e.g. to save power)
- Link bandwidth/loss/delay asymmetry
- Heterogeneous Network Architectures
- Protection of high-value assets
- Limited Emission Requirements (LPI/LPD)

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… ouch
- Maximum size: 64KB (or 4GB… ouch again)
- Fragmentation function:
  - Round to nearest 8 byte boundary
  - Whole datagram lost if any fragment lost… ouch
  - 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
- 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)

IP Routing May Not Work [2]

- Routing protocol performs poorly in environment
  - Topology discovery dominates capacity
  - Incompatible topology assumptions
    - OSPF broadcast model for MANETs
  - Insufficient host resources
    - routing table size in sensor networks
  - Assumptions made of underlying protocols
    - BGP's use of TCP

What about UDP?

- UDP preserves application-specified boundaries
  - May result in frequent fragmentation
  - Permits out-of-order delivery (no sequencing)
- Delay insensitive [no timers]
  - No provision for loss recovery
- No control loops
  - No flow/congestion control or loss recovery
- Works in simplex/bcast/mcast environment
  - no connections

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

Architecture Implications of PEPs [2]

- 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)

What about DNS?

- Names and the DNS:
  - Names: Administrative assignment (global hierarchy)
  - DNS Distributed Lookup Service
    - Name service frequently located near target
    - Requires ~1RTT or more to perform first mapping
    - Caching helps after that
    - Often a reverse-lookup is also required
- Zone updates (TCP)
- Dynamic Updates
- DNS Resolution Failure results in effective application failure or large application delays

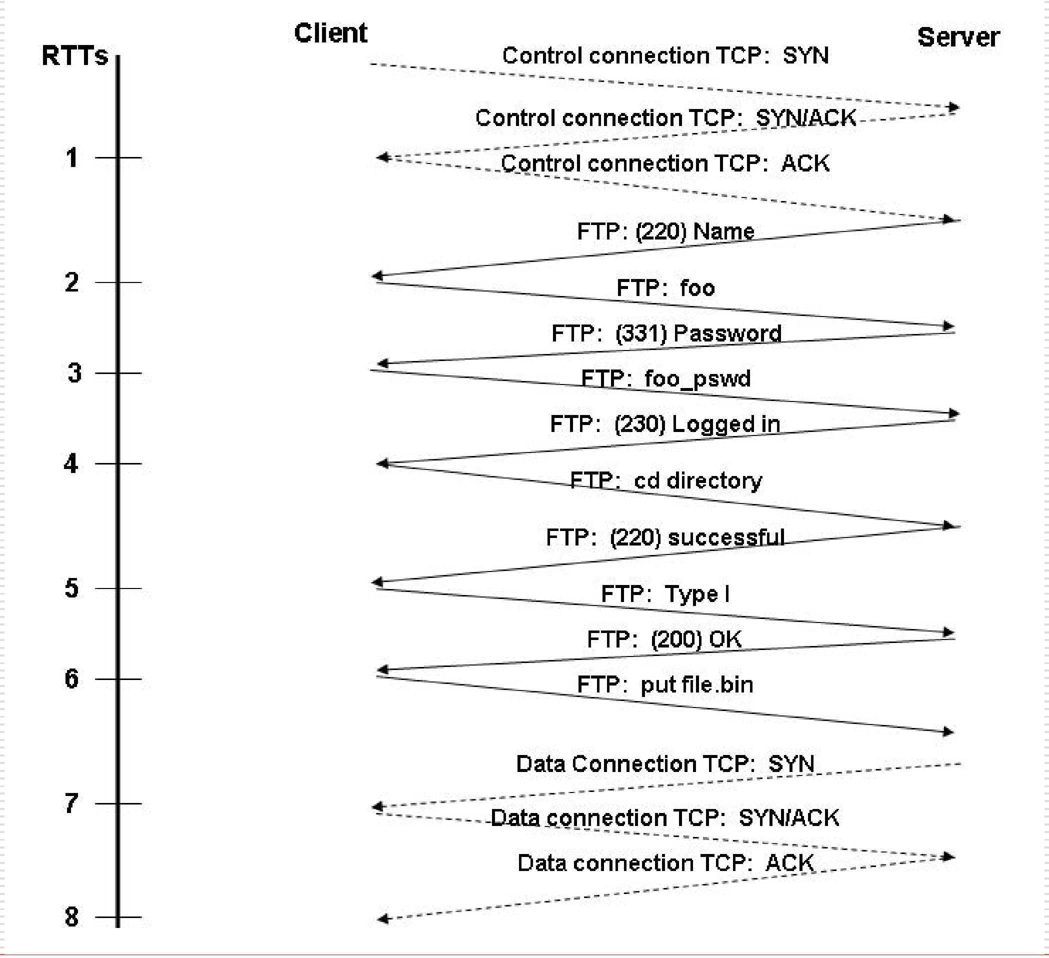
DNS: One level deeper

- "Typical" configuration:
  - Local DNS "close" to client (on the near side of the bad connectivity?)
  - Client typically makes "recursive" call to local DNS: local DNS provides "one stop shopping" for name resolution on behalf of the client
- Local DNS server
  - If address is cached, returns the cached copy
  - Else performs separate *iterative* queries on behalf of the client
    - First, to server that is authoritative for local domain – if there, is returned and we're done; if not responds with list of authoritative servers for TLD of requested name
    - Next, to authoritative server for TLD of requested name – if there, is returned and we're done; if not, responds with authoritative servers for second-level domain
    - Process repeats until IP address is found for requested name
  - Resolved address returned to client
- Issues
  - (Multiple) iterative queries across "challenged" networks
  - Location and configuration of DNS servers for nodes in the "challenged" areas

What about Applications?

- Most use TCP… ouch
- Detecting failures
  - Many applications have an inactivity timeout used to initiate failure-handling
  - Handling failures often means giving up
- Chattiness
  - Many applications implement layer 7 protocols that require lots of round-trip exchanges
  - Extreme cases drive conversation to stop-and-wait
- Robustness to long delays
  - Most apps aren't prepared to continue effectively after re-start or other network disruption
  - And its even worse now with VPNs, NATs, etc.

FTP: An example application



Applications that are interactive exacerbate channel access problems

credit: MITRE

Challenged Networks Roll Call

- Mobile nodes that suffer disruption
  - cell phones, MANETs
- Sensor Networks
  - ZebraNet, mules, etc
- Deep Space Network
- Acoustic underwater networks
- Sneaker nets

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.
  - no really standardized proxy architecture
- Some environments *never* have an e2e path

Outline

- ☑ *Challenged Networks and the Internet Architecture*
- ☐ ***DTN Architecture Overview***
- ☐ *DTN People & Projects*
- ☐ *DTN Research Summary*
- ☐ *DTN Reference Implementation*

15 Minute Break

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

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

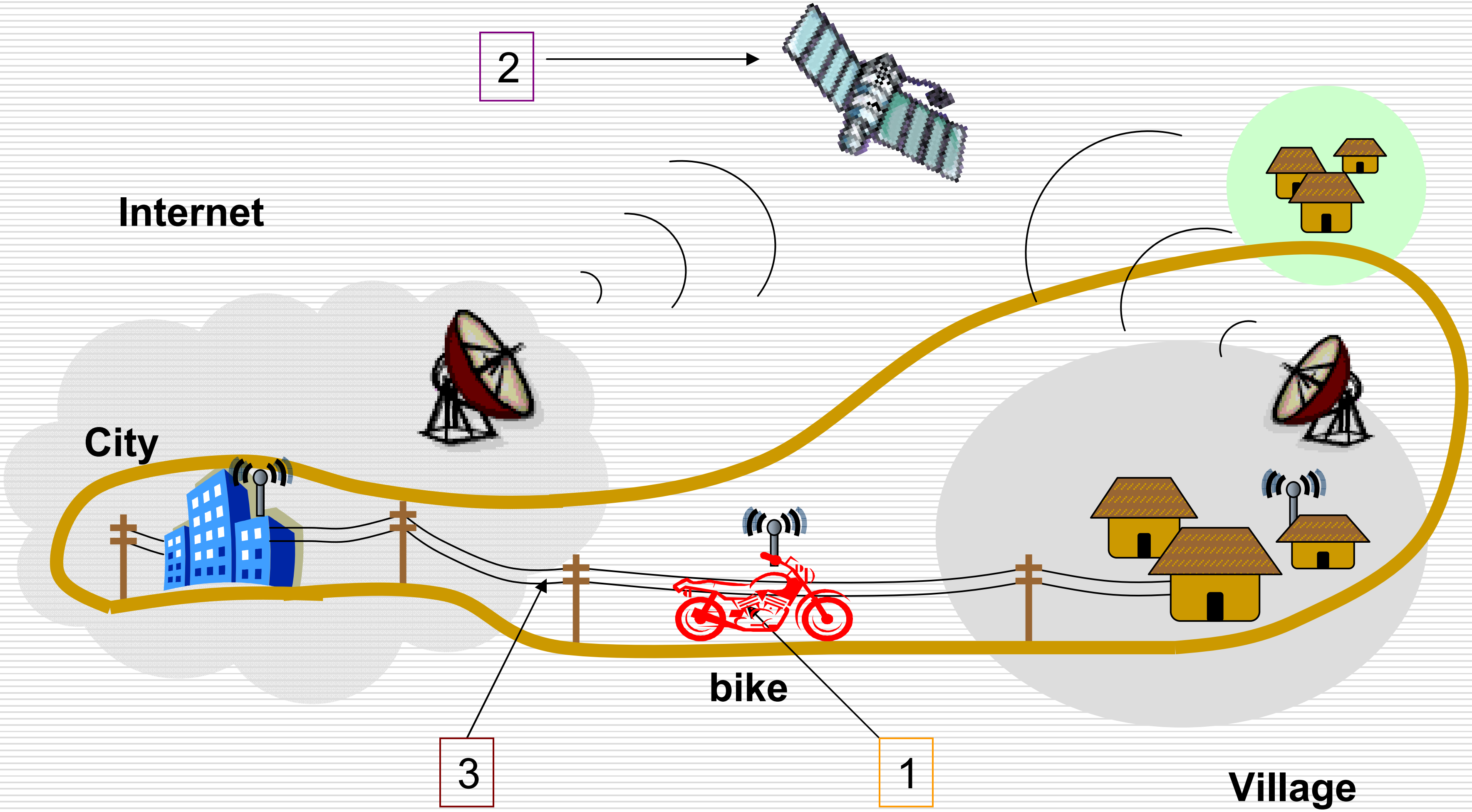
Message Abstraction

- 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

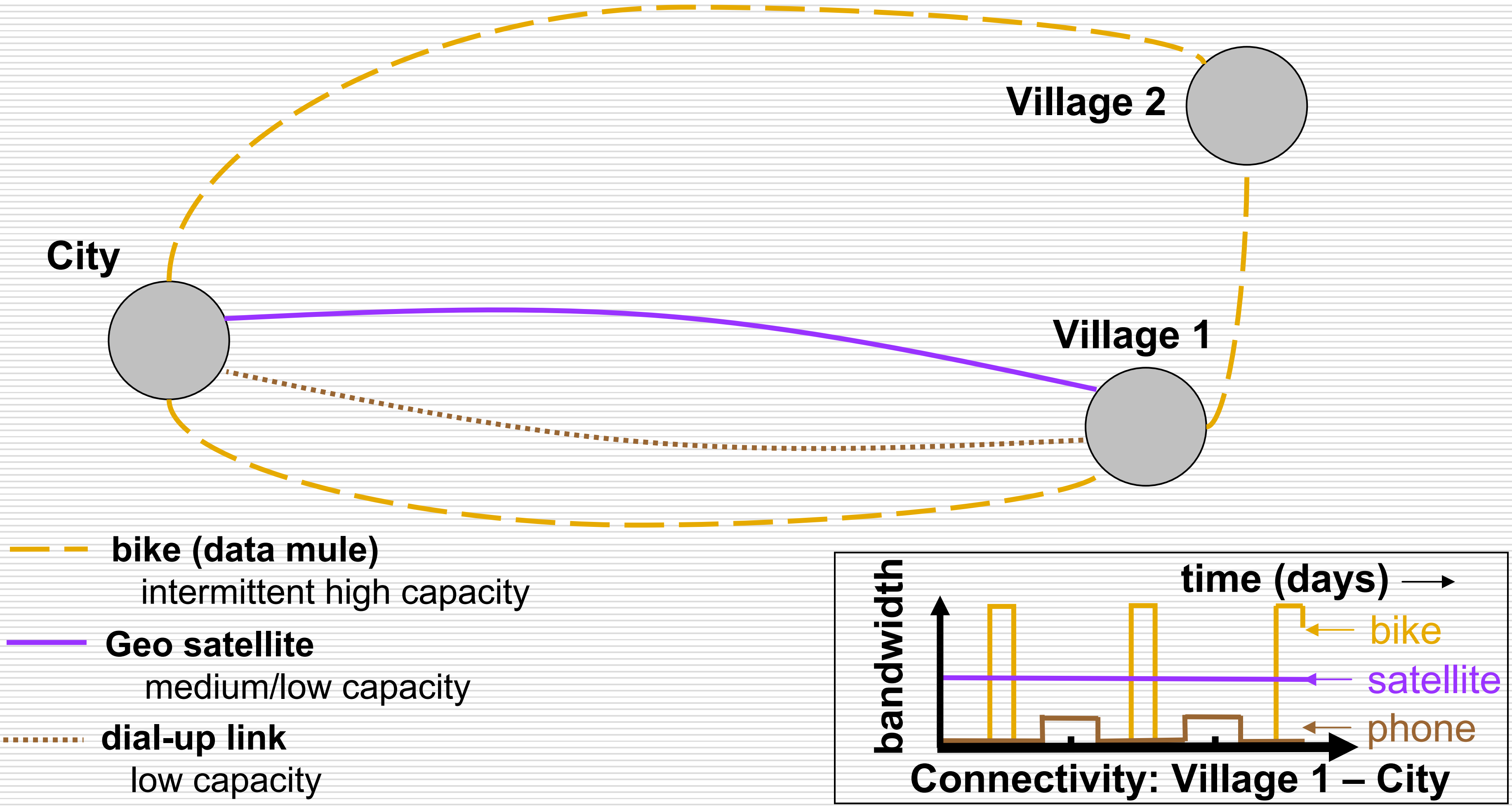
DTN Routing

- 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



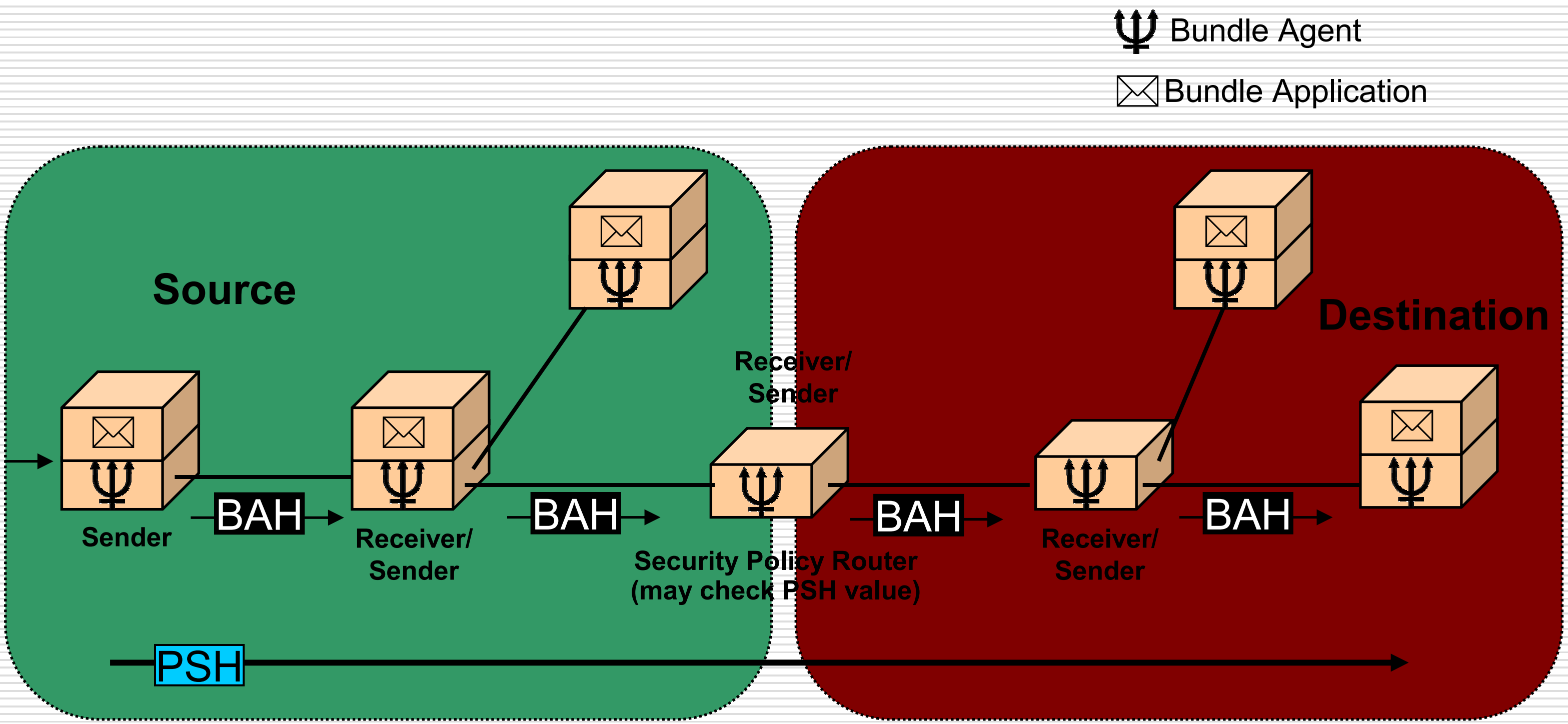
Example Graph Abstraction



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?

DTN Security



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

credit: MITRE

So, is this just e-mail?

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

- Many similarities to (abstract) e-mail service
- Primary difference involves routing, reliability and security
- 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

Outline

- ☑ *Challenged Networks and the Internet Architecture*
- ☑ *DTN Architecture Overview*
- ☐ ***DTN People & Projects***
- ☐ *DTN Research Summary*
- ☐ *DTN Reference Implementation*

DTN People & Projects

- Intel Research – Kevin Fall, Michael Demmer
- UCB – Eric Brewer, Bowei Du
- UCSB – Kevin Almeroth, Khaled Harras
- USC – Thrasyvoulos Spyropoulos, Konstantinos Psounis, Cauligi Raghavendra
- Trinity (Ireland) – Stephen Farrell
- Ohio – Mani Ramadas
- HUT (Finland) – Jörg Ott
- Luleå (Sweden) – Anders Lindgren, Avri Doria
- Waterloo – S. Keshav, Darcy
- Univ. of Massachusetts Amherst – Brian Levine
- Nottingham (UK) – Milena Radenkovic

DTN People & Projects [2]

- BBN – Rajesh Krishnan, Stephen Polit, Ram Ramanathan, Prithwish Basu, David Montana, Vikas Kawadia, Joanne Mikkelson, Regina Rosales Hain, Matthew Condell, Talib Hussain, Mitch Tasman, Partha Pal, Daria Antonova
- JPL – Scott Burleigh, Leigh Torgerson, Esther Jennings, Adrian Hooke
- Google – Vint Cerf
- MITRE – Bob Durst, Keith Scott, Susan Symington, Salil Parikh, Jeff Bush
- SPARTA – Howard Weiss, Sandy Murphy
- Lehigh – Mooi Choo Chuah
- … a few others …

Outline

- ☑ *Challenged Networks and the Internet Architecture*
- ☑ *DTN Architecture Overview*
- ☑ *DTN People & Projects*
- ☐ ***DTN Research Summary***
- ☐ *DTN Reference Implementation*

DTN Research

- Selected Research papers
  - SIGCOMM 2003– the architecture
  - SIGCOMM 2004– routing in DTN
  - SIGCOMM 2005– use of erasure coding
  - Infocom 2005/6– vehicle routing
  - NPSEC 2005– security based on HIBC
  - Milcom 2005– performance and proxies
- Conferences & Workshops
  - SIGCOMM/WDTN 2005
  - ICWN/DTN 2005
  - SIGCOMM/CHANTS 2006
  - CoNext 2006
  - IWCMC/DTMN 2006

IRTF Documents

- draft-irtf-dtnrg-arch – the architecture
- draft-irtf-dtnrg-bundle-security– security protocols
- draft-irtf-dtnrg-bundle-spec– base bundle protocol
- draft-irtf-dtnrg-ltp– high-delay transport protocol
- draft-irtf-dtnrg-ltp-extensions– options for LTP
- draft-irtf-dtnrg-ltp-motivation– why LTP?
- draft-irtf-dtnrg-sec-overview– security summary
- see https://datatracker.ietf.org

DTN Architecture Definition

- Defined architecture goals
  - Interoperability across architectures
  - Reasonable performance in high loss/delay and frequently-disconnected environments
- Components
  - Flexible Naming Scheme with late binding
  - Message Based Overlay Abstraction and API
  - Routing and link/contact scheduling w/CoS
  - Per-hop Authentication and Reliability
- Routing problem formulation as LP

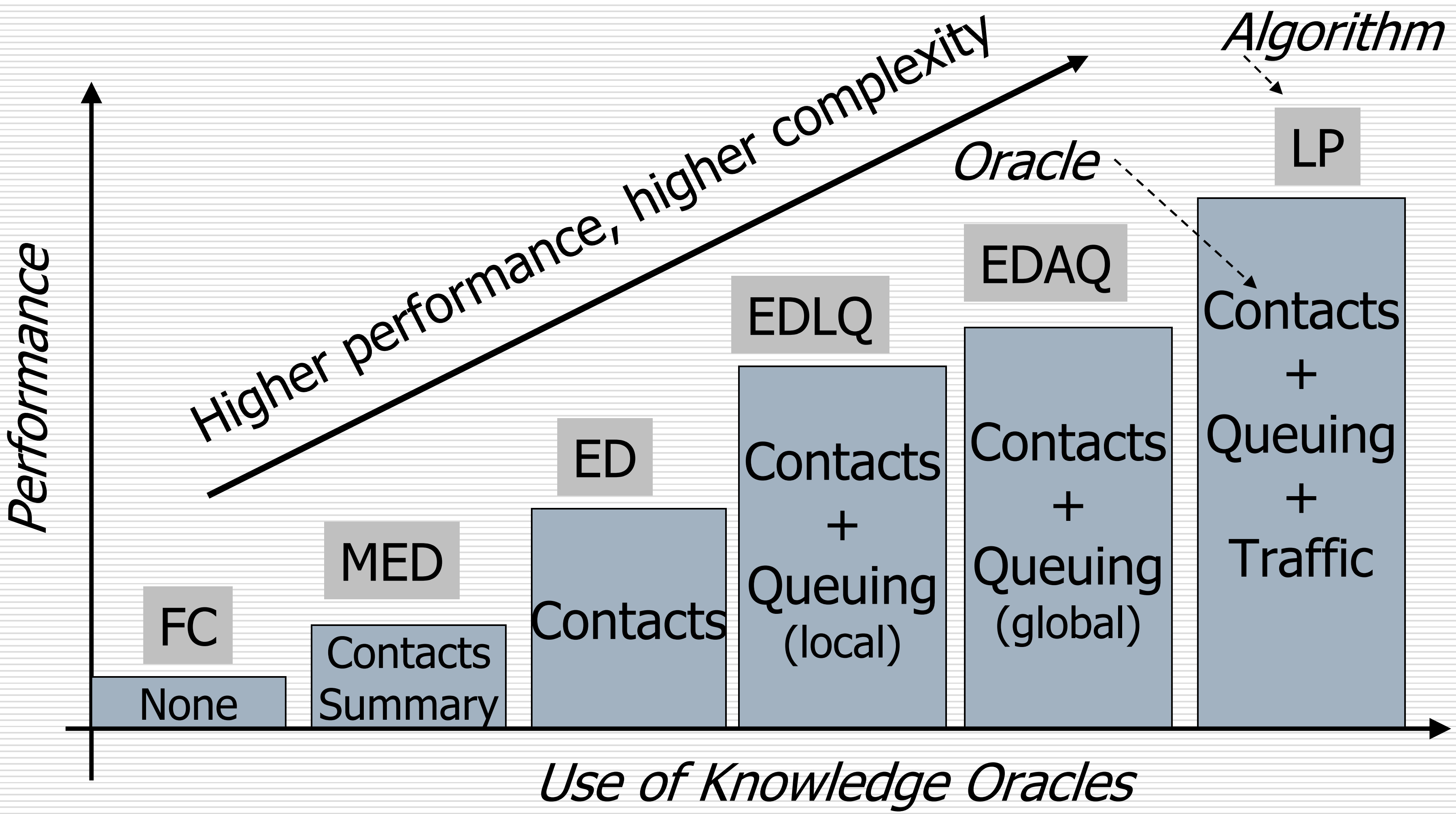
K. Fall, SIGCOMM 2003

DTN Routing

- Routing problem formulation
  - Network as a time-variant multigraph with defined delay / capacity / storage limits
  - Objective: *Minimize average delay*
- Comparison of routing algorithms
  - "oracles" with varied knowledge about contacts, queuing, traffic
- Simulation results
  - Model village access network with LEO satellite, motorbike, and periodic dialup

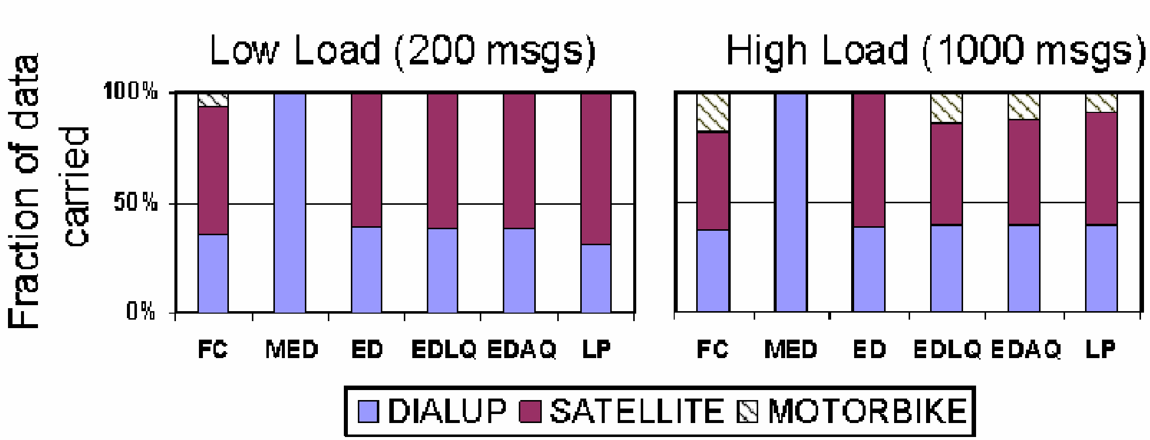
S. Jain, K. Fall, R. Patra – SIGCOMM 2004

Knowledge-Performance Tradeoff



Slide by Sushant Jain

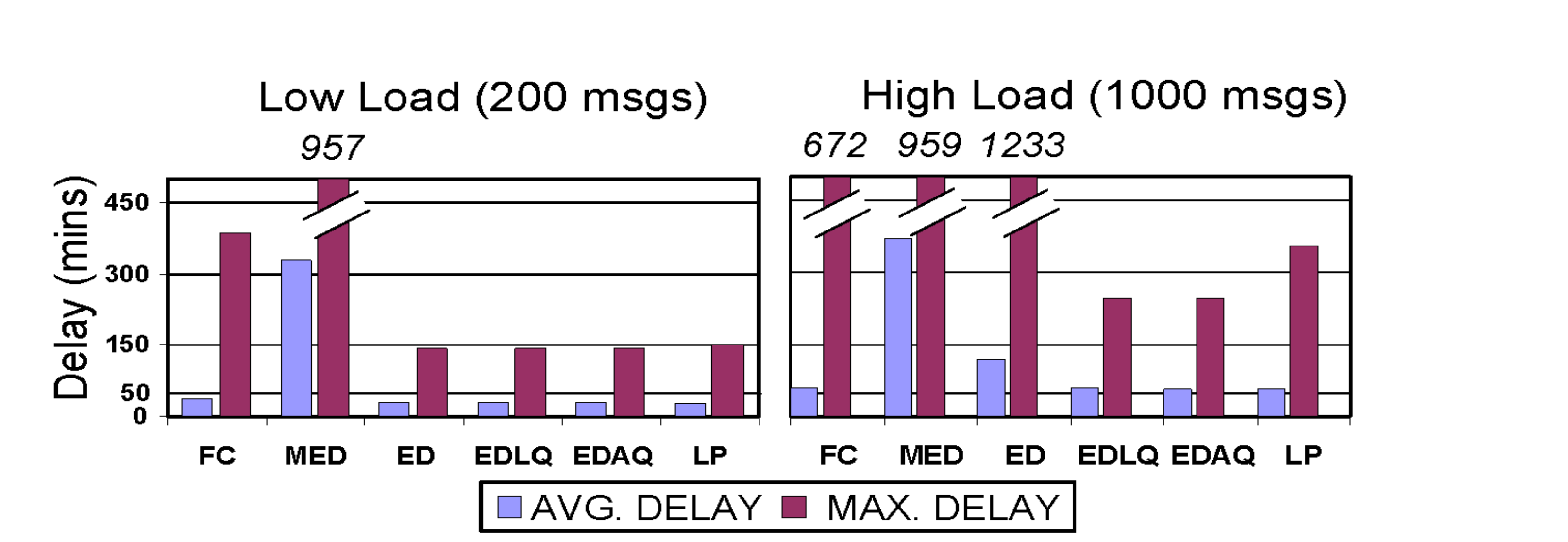
Data Allocations by Algorithm



Min Expected Delay (MED): All data is carried by dialup
Earliest Delivery (ED): Same for low and high load.
{Split between dialup and satellite}
ED, EDLQ, EDAQ make same choices for low load
EDLQ, EDAQ start to use bike also

Slide by Sushant Jain

Delivery Delay Comparison



Low load: ED, EDLQ, EDAQ approx. same performance
High load: EDLQ, EDAQ are optimal. ED is much worse
MED has high delay in both cases
FC performs well on average delay
but has much worse max delay

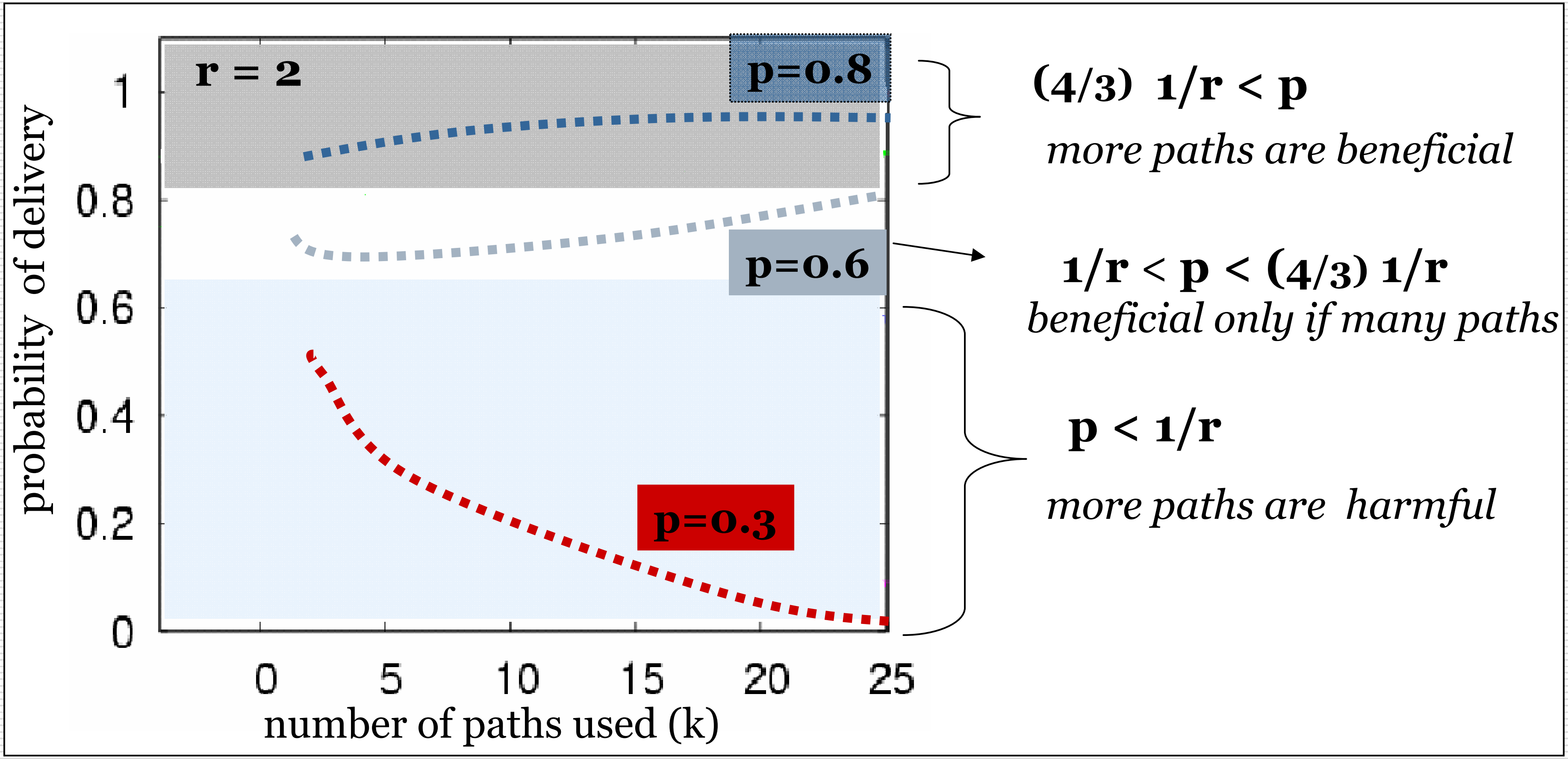
Slide by Sushant Jain

DTN Routing with Failures

- Consider problem of how to transmit bundles over links of different reliability
  - Erasure coding vs. Simple Fragmentation
  - Varied block allocation algorithms
  - Optimal Integer Programming formulation
- Simulation Evaluation
  - Simple case of IID links
  - More complex examples with dependencies

S. Jain, M.Demmer, R. Patra, K. Fall – SIGCOMM 2005

Simple Scenario Results



Slide by Sushant Jain

Portfolio Based Allocation Algorithm

Mapping to the stock portfolio management problem

path	stocks
success probabilities	stock returns
code-blocks allocation	investment portfolio
probability of delivery	probability of achieving a threshold wealth

Markowitz Allocation Algorithm:

allocation on path i $\propto \dfrac{p_i - (1/r)}{(1-p_i)\,p_i}$

average goodness
----
variance

Slide by Sushant Jain

DieselNet & MaxProp (UMass Amherst)

- Opportunistic Routing Protocol
  - scheduling based on likelihood of delivery
  - packets with low hop-counts get high priority
  - congestion -> delete in reverse order
  - acks / anti-packets delivered globally
  - hoplists prevent duplication
- Results
  - better than likelihood along, random or oracle
- DieselNet Testbed
  - buses around Amherst
  - throwboxes (mote + stargate)
- http://prisms.cs.umass.edu/diesel

Disconnected Security (Waterloo)

- Security for disconnected nodes… Problems:
  - secure opportunistic channel establishment
  - mutual opportunistic authentication
  - protection from overrun entities
  - PKI works poorly if connectivity is poor
- Approach using hierarchical Identity Based Crypto
  - IBC: generate public key based on a string but private key must be generated by private key generator (PKG)
  - HIBC: cooperating hierarchy of PKG's
  - no lookup required to find disconnected node's pkey

Disconnected Security [2]

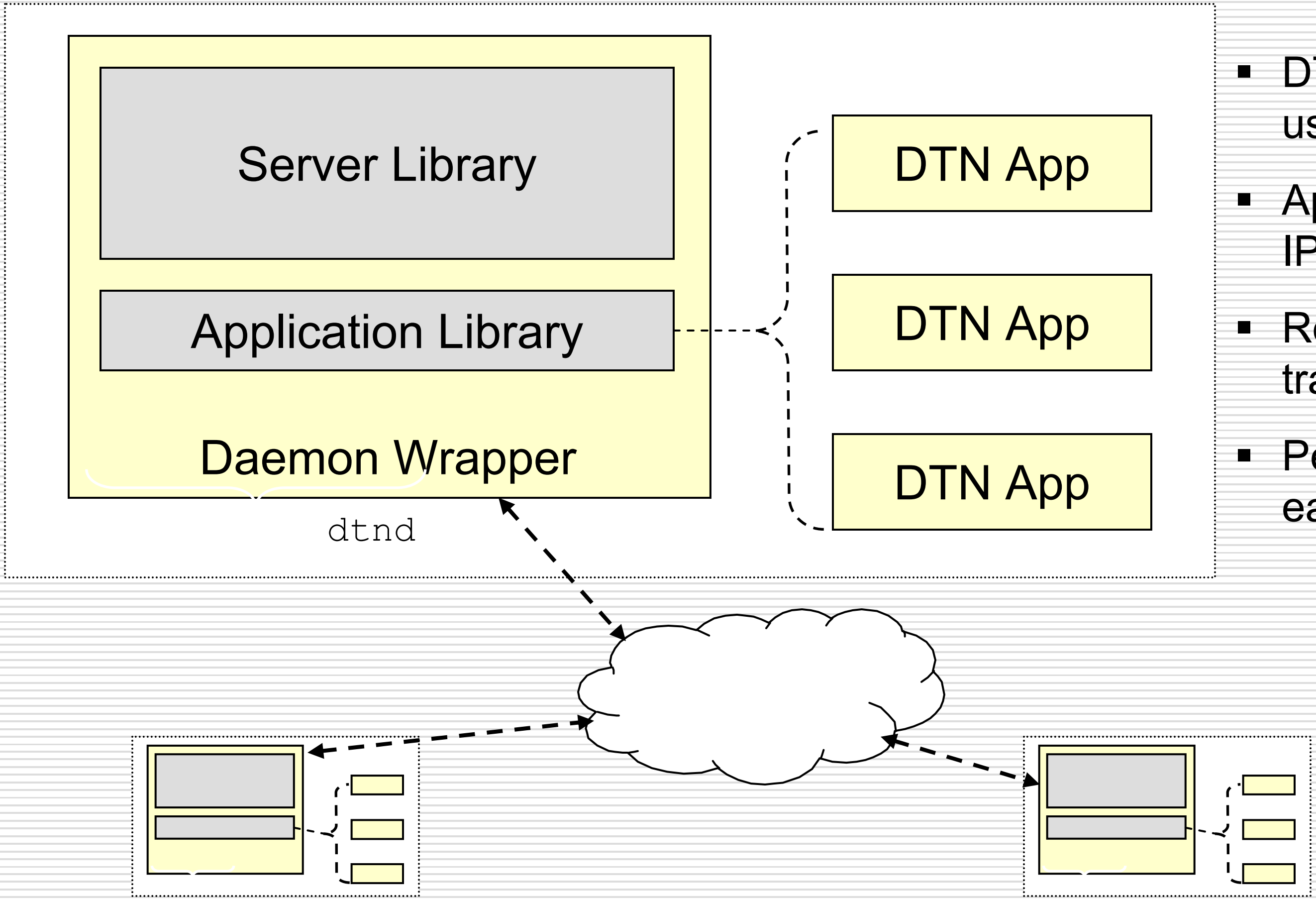
- Bootstrap
  - new user communicates w/PKG over secure channel to get initial key pair
  - can also used tamper-resistant device
  - reversal of accumulated source route used for PKG to reach new node
- Use of Time
  - add datestamp to public key ID's helps to minimize compromise time if device is lost
  - time-based keys instead of CRLs
    - fail-safe versus fail-insecure (CRLs)
- http://blizzard.cs.uwaterloo.ca/tetherless

Outline

- ☑ *Challenged Networks and the Internet Architecture*
- ☑ *DTN Architecture Overview*
- ☑ *DTN People & Projects*
- ☑ *DTN Research Summary*
- ☐ ***DTN Reference Implementation***

15 Minute Break

DTN Reference Implementation

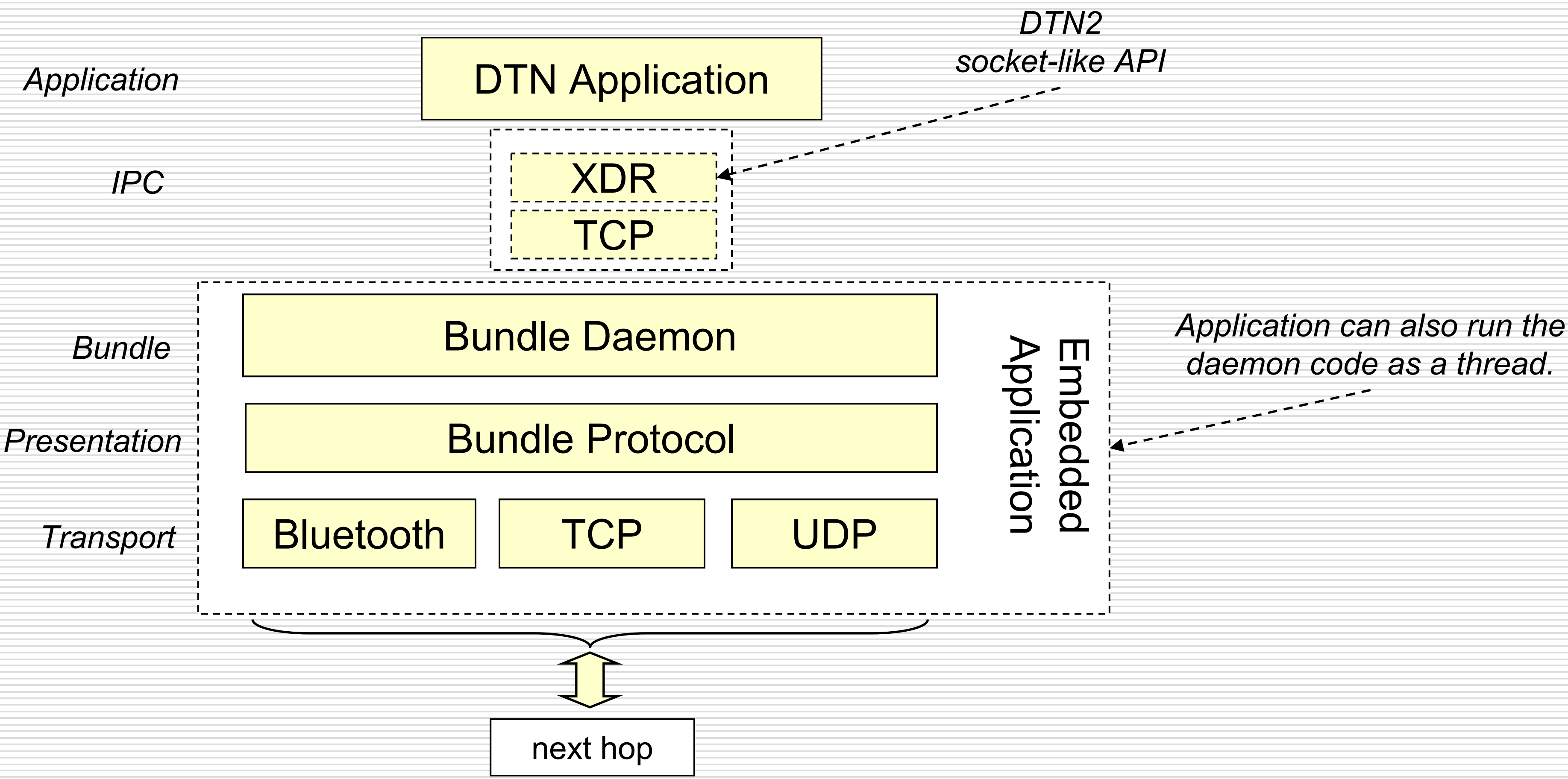


- DTN Router runs as a userspace daemon
- Applications interact via IPC-based API
- Routers use various transport networks
- Persistent storage at each hop in the net

Implementation Details

- Written primarily in C++
  - ~23K non-comment lines of C++ (~4,200 C)
  - ~20K more in generic system support classes (oasys)
  - 154 dtn classes, 201 oasys classes
  - Multithreaded (pthreads), mutex, spin lock
  - STL for data structures (string, list, map, …)
- Design emphasizes clarity, cleanliness, flexibility
- Ported to Linux, Solaris, Win32 (Cygwin), Linux on PDA (ARM), FreeBSD, Mac OSX

ISO Stack View



(slide thanks to Salil Parikh, MITRE)

Implementation Features

- Embedded Tcl Interpreter
  - Configuration parser, admin interface
  - Test script library for verification
- Flexible persistent storage interface
  - Berkeley DB, Filesystem
- Internal API for extensions
  - Convergence Layers, Routers, etc

Terminology

- *Bundle:* Application specified data message
- *Link:* Connection abstraction to next-hop DTN router
- *Interface:* Abstraction that listens for bundles to be received at the daemon
- *Convergence Layer*: Transport-specific implementation of link/interface
- *Endpoint*: One (or more) nodes that are intended to receive a bundle
- *Endpoint ID:* URI name for an endpoint
- *Route:* Maps an endpoint id pattern to a link along with options for the given route

Naming and Addressing

- URI format for names
  - (scheme:scheme-specific-part)
- Extensible scheme support
- dtn scheme pending registration

Scheme	Scheme Specific Part
dtn	dtn://<node>/<demux>
mailto	mailto:demmer@cs.berkeley.edu
eth	eth:00:0d:93:ff:fe:2e:f1:90
wildcard	*

Examples	
Bundle Destination	dtn://sandbox.dtnrg.org.dtn/dtnping.5010
Null Endpoint ID	dtn:none
RouteTable (destination pattern)	dtn://sandbox.dtnrg.org.dtn/*
RouteTable (default pattern)	*:*

Configuration

```
console set addr 127.0.0.1
console set port 5050
```

```
interface add iface-udp udp
interface add iface-tcp0 tcp \
     local_addr=192.168.1.2
interface add iface-tcp1 tcp \
     local_addr=10.1.1.1
```

```
storage set type berkeleydb
storage set dbdir /var/dtn
storage set dbname DTN
storage set payloaddir \
   /var/dtn/bundles
```

```
param set accept_custody true
param set reactive_frag_enabled true
param set link_max_retry_interval 300
```

```
link add link-larry larry:5000 ONDEMAND
tcp
link add link-moe moe:5000 ALWAYSON udp
link add link-moe2 moe:5001 ALWAYSON tcp
```

```
route set type static
route set local_eid dtn://curly.dtn

route add dtn://larry.dtn/* link-larry
route add dtn://moe.dtn/* link-moe
route add dtn://* link-larry priority=-1
```

Console Interface

```
dtn% help
For help on a particular command, type "help <cmd>".
The registered commands are:
        api bundle console debug help interface link log param registration
        route shutdown storage test

dtn% help route
route set type
        Which routing algorithm to use.
route add <dest> <link/endpoint> [opts]
        add a route
route del <dest> <link/endpoint>
        delete a route
route dump
        dump all of the static routes

dtn% route dump
Route table for static router:
        dtn://jitara.demmer.nu.dtn/* -> link-jitara (FORWARD_COPY) priority 0
[custody timeout: base 1800 lifetime_pct 25 limit 0]
Links:
OPPORTUNISTIC link-jitara -> jitara-192.demmer.nu:5000 (UNAVAILABLE)
```

Debug Logging System

- Hierarchical logging targets
- Logging Levels: critical, error, warning, notice, info, debug

~/.dtndebug file:

```
/ notice
/dtn/bundle/daemon info
/dtn/cl/tcp debug
/dtn/cl/tcp/listener info
```

```
[1147557395.879452 /dtnd notice] DTN daemon starting up... (pid 930)
[1147557395.930501 /dtn/cl/tcp debug] adding interface tcp0
[1147557395.930890 /dtn/cl/tcp/iface/tcp0 debug] created socket 18
[1147557395.930920 /dtn/cl/tcp/iface/tcp0 debug] setting SO_REUSEADDR
[1147557395.930956 /dtn/cl/tcp/iface/tcp0 debug] binding to 127.0.0.1:10002
[1147557395.931025 /dtn/cl/tcp/iface/tcp0 debug] listening
[1147557395.931076 /dtn/cl/tcp/iface/tcp0 debug] state INIT -> LISTENING
[1147557395.931462 /dtn/cl/tcp debug] adding ONDEMAND link localhost:11002
[1147557397.401413 /dtn/bundle/daemon info] REGISTRATION_ADDED 0 dtn://host-0
[1147557397.401979 /dtn/bundle/daemon notice] loading bundles from data store
[1147557397.402419 /dtn/bundle/daemon info] LINK_AVAILABLE ONDEMAND tcp:0-1 ->
localhost:11002 (AVAILABLE)
[1147557401.382403 /dtn/cl/tcp/iface/tcp0 debug] accepted connection fd 29 from 127.0.0.1:50576
[1147557401.382490 /dtn/cl/tcp/iface/tcp0 debug] new connection from 127.0.0.1:50576
[1147557401.382692 /dtn/cl/tcp/conn/127.0.0.1:50576/29 debug] setting SO_REUSEADDR
[1147557401.382885 /dtn/cl/tcp/conn/127.0.0.1:50576 debug] connection main loop starting up...
[1147557401.382928 /dtn/cl/tcp/conn/127.0.0.1:50576 debug] accept: sending contact header...
[1147557401.383075 /dtn/cl/tcp/conn/127.0.0.1:50576/29 debug] ::writev() fd 29 cc 12
[1147557401.383119 /dtn/cl/tcp/conn/127.0.0.1:50576/29 debug] writeall 12 bytes 0 left 12 total
```

Application Interface

- IPC implementation over loopback TCP
  - XDR structures used for data transfer
- Bundle data passed to/from the daemon in memory or through a local file
- Hooks to manipulate persistent registrations (akin to listening sockets)
- Basic send/recv interface for bundles
- Polling hooks to integrate with application event loop

API Example Pseudocode

Send a bundle to dest_eid:

```
h = dtn_open()

dtn_build_local_eid(h, &local_eid,
                    “app_string”)

bundle_spec.source = local_eid
bundle_spec.dest   = dest_eid
bundle_spec.expiration = 60 * 30;

dtn_set_payload(&payload,
                DTN_PAYLOAD_MEM,
                “test payload”, 12);

dtn_send(h, &bundle_spec, &payload)

dtn_close(h)
```

Receive a bundle for dest_eid:

```
h = dtn_open()

reginfo.endpoint = dest_eid
reginfo.expiration = 30
reginfo.failure_action =
DTN_REG_DEFER

dtn_register(h, reginfo, &regid)
dtn_bind(h, regid)

dtn_recv(h, &bundle_spec, &payload,
         DTN_PAYLOAD_MEM, -1)

dtn_unregister(h, regid)

dtn_close(h)
```

Application Interface Details

```
dtn_handle_t dtn_open();

int dtn_close(dtn_handle_t handle);

int dtn_errno(dtn_handle_t handle);

char* dtn_strerror(int err);
```

```
int dtn_register(dtn_handle_t handle,
                 dtn_reg_info_t info,
                 dtn_reg_id_t* id);

int dtn_unregister(dtn_handle_t handle,
                   dtn_reg_id_t* id);

int dtn_bind(dtn_handle_t handle,
             dtn_reg_id_t regid);

int dtn_unbind(dtn_handle_t handle,
               dtn_reg_id_t regid);
```

```
int dtn_send(dtn_handle_t handle,
             dtn_bundle_spec_t* spec,
             dtn_bundle_payload_t* payload);

int dtn_recv(dtn_handle_t handle,
             dtn_bundle_spec_t* spec,
             dtn_bundle_payload_t* payload,
             dtn_bundle_payload_location_t l,
             dtn_timeval_t timeout);

int dtn_begin_poll(dtn_handle_t handle,
                   dtn_timeval_t timeout);

int dtn_cancel_poll(dtn_handle_t handle);
```

Application: dtnsend

- Basic bundle transmission application
- Payload specified by file or command line
- Supports options for class of service, custody transfer, status reports

Application: dtnsend usage

```
% dtnsend/dtnsend -h
usage: dtnsend/dtnsend [opts] -s <source_eid> -d <dest_eid> -t <type> -p <payload>
options:
 -v verbose
 -h help
 -s <eid|demux_string> source eid)
 -d <eid|demux_string> destination eid)
 -r <eid|demux_string> reply to eid)
 -t <f|m|d> payload type: file, message, or date
 -p <filename|string> payload data
 -e <time> expiration time in seconds (default: one hour)
 -i <regid> registration id for reply to
 -n <int> copies of the bundle to send
 -z <time> msecs to sleep between transmissions
 -c request custody transfer
 -C request custody transfer receipts
 -D request for end-to-end delivery receipt
 -R request for bundle reception receipts
 -F request for bundle forwarding receipts
 -w wait for bundle status reports
```

Application: dtnrecv

- Primarily a testing application
- Support for registration manipulation
- Prints a hexdump of payload:

```
% apps/dtnrecv/dtnrecv dtn://test.dtn/dest -n 1
dtnrecv (pid 467) starting up -- count 1
register succeeded, regid 13
binding to regid 13
dtn_recv [dtn://test.dtn/dest]...
30 bytes from [dtn://test.dtn/source]: transit time=0 ms
0000000 7468 6973 2069 7320 736f 6d65 2074 6573 |  this is some tes
0000010 7420 7061 796c 6f61 6420 6461 7461      |  t payload data

dtnrecv (pid 467) exiting: 1 bundles received, 30 total bytes
```

Application: dtnrecv usage

Terminal

```
% apps/dtnrecv/dtnrecv -h
usage: apps/dtnrecv/dtnrecv [opts] <endpoint>
options:
 -v verbose
 -q quiet
 -h help
 -d <eid|demux_string> endpoint id
 -r <regid> use existing registration regid
 -n <count> exit after count bundles received
 -e <time> registration expiration time in seconds (default: one hour)
 -f <defer|drop|exec> failure action
 -F <script> failure script for exec action
 -x call dtn_register and immediately exit
 -c call dtn_change_registration and immediately exit
 -u call dtn_unregister and immediately exit
 -N don't try to find an existing registration
 -t <timeout> timeout value for call to dtn_recv
```

Application: dtnping

- Tool to test connectivity to dtn overlay routers
- Uses unspecified ADMIN_ECHO option

Terminal

```
% apps/dtnping/dtnping -h
usage: apps/dtnping/dtnping [-c count] [-i interval] [-e expiration] eid

% apps/dtnping/dtnping dtn://sandbox.dtnrg.org.dtn -c 4
source_eid [dtn://sandbox.dtnrg.org.dtn/ping.15989]
replyto_eid [dtn://sandbox.dtnrg.org.dtn/ping.15989]
dtn_register succeeded, regid 10
checking for bundles already queued...
PING [dtn://sandbox.dtnrg.org.dtn/]...
10 bytes from [dtn://sandbox.dtnrg.org.dtn/]: 'dtn_ping!' time=29.96 ms
10 bytes from [dtn://sandbox.dtnrg.org.dtn/]: 'dtn_ping!' time=6.38 ms
10 bytes from [dtn://sandbox.dtnrg.org.dtn/]: 'dtn_ping!' time=6.13 ms
10 bytes from [dtn://sandbox.dtnrg.org.dtn/]: 'dtn_ping!' time=6.22 ms
```

Application: dtnperf

- DTN end-to-end performance testing app
- Client and server components
  - Bundles flow client to server, status reports returned
  - Tests round trip times

Application: dtnperf usage

```
% dtnperf/dtnperf-client -h

SYNTAX: dtnperf/dtnperf-client -d <dest_eid> [-t <sec> | -n <num>] [options]

where:
 -d <eid> destination eid (required)
 -t <sec> Time-Mode: seconds of transmission
 -n <num> Data-Mode: number of MBytes to send
Options common to both Time and Data Mode:
 -p <size> size in KBytes of bundle payload
 -r <eid> reply-to eid (if none specified, source tuple is used)
Data-Mode options:
 -m use memory instead of file
 -B <num> number of consecutive transmissions (default 1)
 -S <sec> sleeping seconds between consecutive transmissions (default 1)
Other options:
 -c CSV output (useful with redirection of the output to a file)
 -h help: show this message
 -v verbose
 -D debug messages (many)
```

Application: dtncat

- DTN analog to netcat
- Two modes:
  - Data from stdin to DTN
  - Data from DTN to stdout (listen mode)
- Future plans to support streaming input / output protocol:
  - <length> <data> <length> <data> …

Application: dtncat usage

```
% apps/dtncat/dtncat -h
To source bundles from stdin:
    usage: apps/dtncat/dtncat [opts] -s <source_eid> -d <dest_eid>
To receive bundles to stdout:
    usage: apps/dtncat/dtncat [opts] -l <receive_eid>
common options:
 -v verbose
 -h/H help
receive only options (-l option required):
 -l <eid> receive bundles destined for eid (instead of sending)
 -n <count> exit after count bundles received (-l option required)
send only options (-l option prohibited):
 -s <eid|demux_string> source eid)
 -d <eid|demux_string> destination eid)
 -r <eid|demux_string> reply to eid)
 -e <time> expiration time in seconds (default: one hour)
 -i <regid> registration id for reply to
 -c request custody transfer
 -C request custody transfer receipts
 -D request for end-to-end delivery receipt
 -R request for bundle reception receipts
 -F request for bundle forwarding receipts
 -w wait for bundle status reports
```

Application: dtntunnel

- Proxy for generic UDP traffic
- TCP support under development
- Used to extending apps to DTN-enabled networks
  - Also to compare DTN vs. traditional protocols

Application: dtntunnel usage

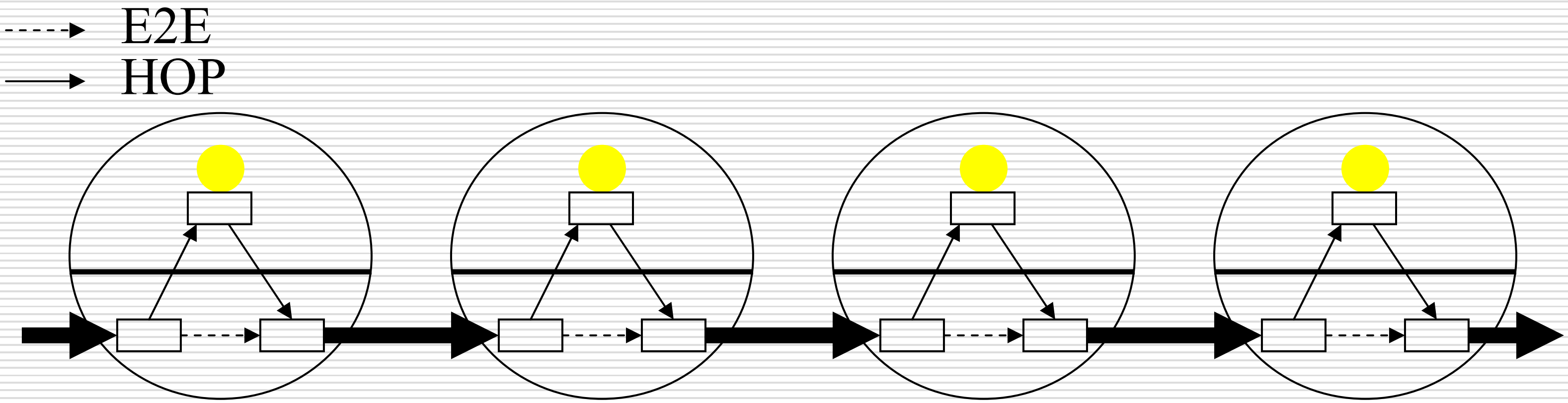
```
% apps/dtntunnel/dtntunnel -h
usage: dtntunnel [opts]

opts:
  -h, --help                 show usage
  -o, --output <output>      file name for logging output (- indicates stdout)
  -l <level>                 default log level [debug|warn|info|crit]
  -L, --listen               run in listen mode for incoming CONN bundles
  -e, --expiration <secs>    expiration time
  -t, --tcp                  proxy for TCP connections
  -u, --udp                  proxy for UDP traffic
  -d, --dest_eid <eid>       destination endpoint id
  --local_eid_override <eid>
                             local endpoint id
  --laddr <addr>             local address to listen on
  --lport <port>             local port to listen on
  --rhost <addr>             remote host/address to proxy for
  --rport <port>             remote port to proxy
  -z, --max_size <bytes>     maximum bundle size for stream transports (e.g. tcp)
```

Application: dtncp / dtncpd

- DTN file transfer application
- Server puts files in per-source directory
- Client waits for return receipt ack

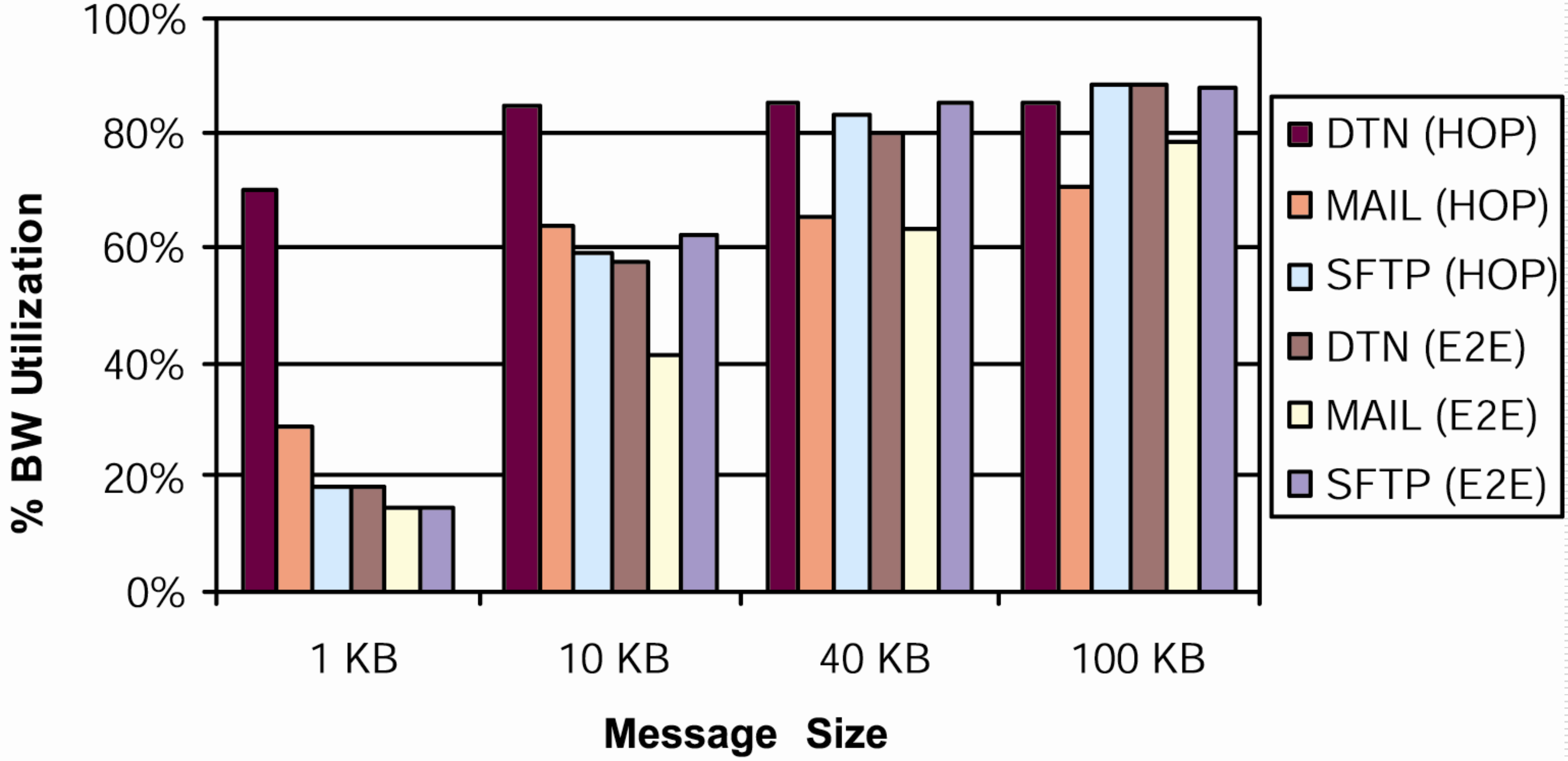
```
% dtncp/dtncp -h
usage: dtncp/dtncp [filename] [destination_eid] [remote-name]
    Remote filename is optional; defaults to the local filename.

% dtncpd/dtncpd -h
usage: dtncpd/dtncpd [ directory ]
    optional directory parameter is where incoming files will get put
    (defaults to: /var/dtn/dtncpd-incoming)
```

Evaluation: Experiment Setup



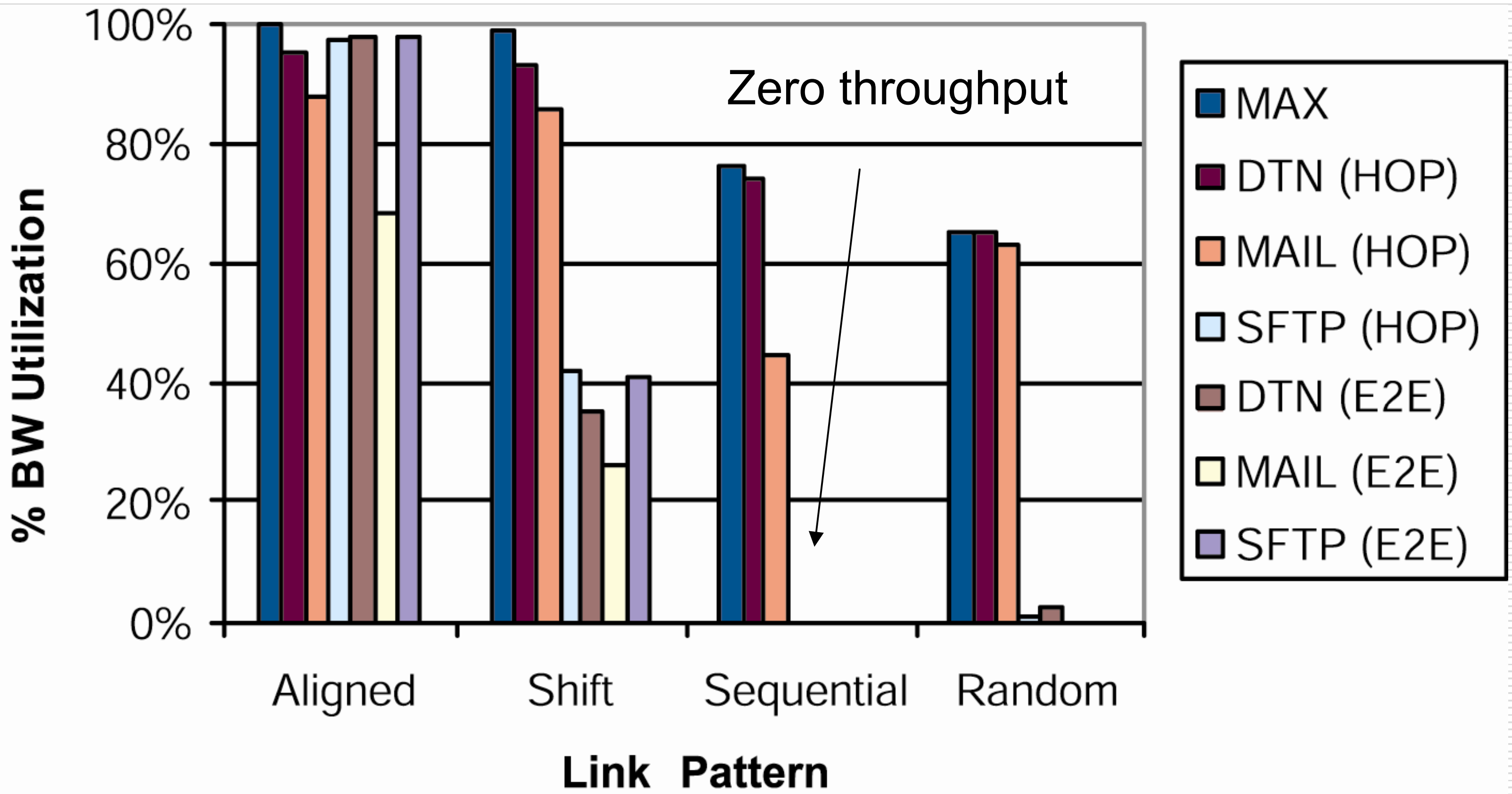
- Compare robustness to interruption / link errors
- Approaches compared
  - End-to-end TCP (kernel routing)
  - Proxied (TCP ‘plug proxies’)
  - Store-and-forward (Sendmail, no ckpoint/restart)
  - DTN (store-and-forward with restart)
- Link up/down patterns: aligned, shifted, sequential, random

Evaluation: BW Efficiency



No disruptions: DTN does well for small msgs, modest overhead overall

Evaluation: Interruption Tolerance



Up/down 1m/3min; 40kb messages; shift: 10s

Availability

- All code is open source and freely available
  - http://www.dtnrg.org/wiki/Code
  - Regular tarball releases
  - Debian packages (stable i386)
  - Anonymous CVS
- dtn-users mailing list
  - http://mailman.dtnrg.org/mailman/listinfo/dtn-users

Major TODO Items

- Full-Fledged routing implementation
- Dynamic Neighbor discovery
- Multi-path forwarding
- Proactive Fragmentation (for real)
- External Router / Storage / etc
- Documentation :-)
- Security integration and testing

Outline

- *Challenged Networks and the Internet Architecture*
- *DTN Architecture Overview*
- *DTN People & Projects*
- *DTN Research Summary*
- *DTN Reference Implementation*

Relevant Links

- DTNRG:
  - http://www.dtnrg.org
- DARPA DTN Program:
  - http://www.darpa.mil/ATO/solicit/DTN/index.htm
- Dieselnet:
  - http://prisms.cs.umass.edu/diesel/
- Tetherless Computing Architecture:
  - http://mindstream.watsmore.net/
- EDIFY Research Group:
  - http://edify.cse.lehigh.edu/
- Technology and Infrastructure for Emerging Regions:
  - http://tier.cs.berkeley.edu/
- Drive-Thru Internet
  - http://www.drive-thru-internet.org/