DISRUPTION TOLERANT NETWORKING PROXIES FOR ON-THE-MOVE TACTICAL NETWORKS

Keith Scott
The MITRE Corporation
McLean, VA

ABSTRACT

A major tenet of Network Centric Warfare is that information sharing is vital to future battlefield superiority. Many plans for implementing NCW envision using IP as the network infrastructure to support information sharing, from wired and satellite networks to tactical networks. Unfortunately, IP implementations depend on the existence of stable end-to-end paths. If a router receives a packet and has nowhere to send it, the packet is dropped.

Many tactical networks exhibit disconnections in connectivity that can severely impair, or completely halt, IP traffic. Disconnection Tolerant Networking (DTN) provides an end-to-end communications service in the presence of network disruptions, including (non-permanent) network partitions.

DTN provides its own application programming interface, which is different from the sockets interface commonly used for TCP/IP. One way to adapt existing applications to use DTN is via application layer proxies. These proxies translate from applications' native use of end-to-end IP-based protocols such as TCP and UDP into DTN messages (bundles) that can traverse disrupted networks. This paper describes how DTN application layer proxies can be used with a CONDOR Jump-C2 class vehicle, using an HTTP (web)-to-DTN proxy as an example.

INTRODUCTION

A. Network Centric Warfare (NCW)

The DoD report to Congress on Network Centric Warfare (NCW) [1] states that "NCW has the potential to increase warfighting capabilities by orders of magnitude." The main tenet of NCW is that information sharing will enhance situational awareness, enable collaboration, and enhance sustainability and speed of command [1]. NCW envisions a rich sharing of information, including imagery, position information, radar information, etc. Many of these information types (e.g. imagery) require reliable delivery.

Unfortunately, the warfighters who most need enhanced situational awareness and can best take advantage of increased operational tempo are also operating in challenging tactical communications environments. Data rates in the low tens of kilobits per second and below are common, and dropouts due to terrain masking, multipath in urban environments, foliage (for high frequencies), and jamming are also common. As an example, Figure 1 shows the message completion rates for UDP messages in a Future Combat Systems (FCS) test [2].

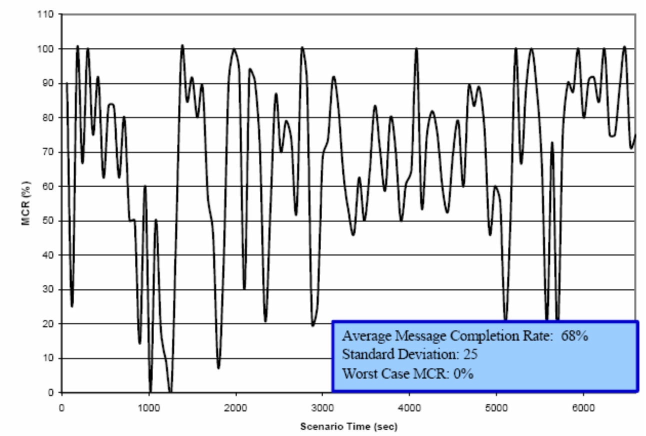


Figure 1: Message Completion Results for FCS

In Figure 1, the average message completion rate of 68% is much too low to support TCP connections, the transport protocol most responsible for reliable data delivery in the Internet. Even relatively loss-tolerant applications such as voice are hard-pressed to perform in such environments.

B. IP In Challenged Environments

Many are looking to the Internet Protocol (IP) to be the underpinning for NCW communications. While IP performs well in connected environments, current implementations are not well-suited to situations where there are even temporary outages in communications.

Even though IP is a store-and-forward technology, the timescale for packet storage is small compared to the duration of most outages in tactical environments. If an IP router cannot forward a packet at the moment it tries to make the decision as to the packet's next hop, it drops the packet. Higher layer protocols such as TCP can provide reliability over IP's best-effort delivery model, but TCP performance suffers greatly if the packet loss percentage gets above four or five percent. Thus if NCW is to succeed, some mechanism for providing secure, reliable communications in tactical environments is needed.

C. Disruption Tolerant Networking

Disruption Tolerant Networking (DTN) [3] defines a store-and-forward, message-switched overlay network that operates over traditional transport layers (e.g. TCP, UDP), as well as directly on top of data link layers (e.g. 802.11). Unlike current IP implementations which require a contemporaneous network path from source to destination, DTN can store messages for arbitrary durations at intermediate nodes pending the availability of a next hop.

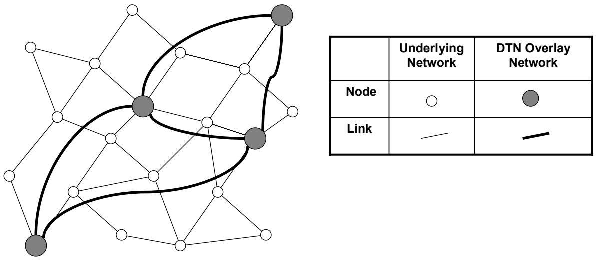


Figure 2: DTN Overlay Network

Figure 2 shows a notional DTN overlay network on top of some underlying infrastructure. Even if the underlying infrastructure is IP-based and running in a lossy tactical environment, DTN will provide reliability for the times when IP drops packets because the routers cannot forward them.

Because messages (bundles, in DTN parlance) move from DTN node to DTN node in the overlay network, they do not have to make it all the way from the source to the destination in one unbroken path. If a bundle arrives at a DTN node and there is no connectivity to the next DTN node in the path, it is simply held until connectivity becomes available (possibly via a new path), or until it times out and expires.

DTN's reliability mechanism depends on *custody transfers* between DTN routers. Each DTN message marked for reliable delivery has one or more custodians whose job is to make sure that the message reaches the destination. As the message flows through the overlay network, custody can be transferred from one DTN router to another. In this way the point from which the message has to be retransmitted if it is lost is moved progressively towards the destination. The main motivation for this is that once a message has made it across a poorly-connected piece of the network, retransmissions don't have to re-cross that piece of network.

DTN is also designed to make efficient use of contention-based tactical radios by reducing the number of back-and-forth interchanges needed to transmit application messages. By using relatively large application messages as the units of transmission and acknowledgement, DTN can reduce the number of channel accesses compared to transferring the message using a packet-based transport layer such as TCP.

Figure 3 illustrates the difference between end-to-end IP-based networking and Disruption Tolerant Networking. Figure 3 shows two depictions of a 5-hop linear network, one using traditional IP (top) and one using DTN (bottom). Time progresses to the right, and connectivity between nodes is indicated by heavy bars on the link timelines and lack of connectivity by thin bars. The source always has data to send, and the heavy bars underneath the link availability bars indicate data transmissions.

The top half of the figure emphasizes IP's need for a contemporaneous end-to-end path before any data can flow, regardless of the transport protocol used (TCP or UDP). If DTN is present at each node in the network, then DTN can store messages at intermediate points. The stored messages can then be forwarded incrementally across the various links. This has the dual advantages of reducing the time to the receipt of the first bit at the destination, as well as increasing the information-carrying capacity of the network.

Results from a four-node experiment using emulab [4] support the conclusions of Figure 3, with DTN providing near maximum throughput regardless of the pattern of link connectivity [5].

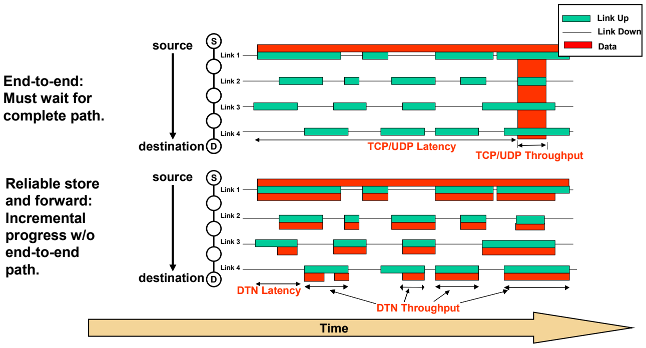


Figure 3: IP requires an end-to-end path; DTN uses individual links as they become available.

Finally, because DTN routers in the overlay can hold on to messages, they can make use of temporal and scheduling information that IP can’t. For example, IP's approach to routing datagrams is to build a picture (via a routing protocol such as RIP, OSPF, IS-IS, etc.) of the current network connectivity, and to route datagrams using only the information about what end systems are reachable at that instant. DTN, on the other hand, is free to use information about what the network will look like in the future (say, if a LEO satellite communications pass is scheduled), or what the network has looked like in the past (a UAV has been visible every day around 1 p.m. for the past week). Thus a DTN router might have a current path through a series of terrestrial wireless networks, but decide to defer transmission of a bundle until an upcoming SATCOM pass.

D. CONDOR

In the early phase of Operation Iraqi Freedom, Marine Corps troops were able to move so quickly that they outran the coverage of their line-of-sight radios [6]. This required them to use other means of communication such as commercial satellite services and satellite telephones to maintain beyond-line-of-sight connectivity. These mechanisms, while expedient, are not something the Marine Corps wants to rely on in the future. The Joint Tactical Radio System (JTRS) is still years away, however, so a near-term solution to connect forward-deployed units is required.

The Marine Corps Command and Control, On-the-Move, Network, Digital, Over-the-horizon Relay (CONDOR) program is seeking to provide connectivity to forward-deployed users by bridging/routing between the forward-deployed units with tactical radios and other users not directly reachable via the tactical radios. The over-the-horizon relay can be provided by any number of technologies, such as INMARSAT, MILSATCOM, Satellite telephone, or UAVs. The CONDOR relay is implemented as a HMMWV with a rack of communications gear, including both terrestrial radio(s) and satellite modem(s).

There are three components of the CONDOR capability set, with the most advanced being a Jump-C2 vehicle providing on-the-move, over-the-horizon communications capabilities to forward-deployed commanders. This paper discusses DTN and application layer proxies for the CONDOR Jump-C2 vehicle to improve performance when the CONDOR's relay connectivity is less than perfect.

DTN SUPPORT FOR APPLICATIONS

Existing applications designed to run over TCP/IP generally use some variant of the Berkeley sockets API, which differs from the DTN API. Thus existing applications don't have access to DTN services. Equally important, applications themselves need to be designed to be delay/disruption tolerant if they hope to make use of DTN. If an application requires numerous message exchanges to accomplish its task, then even if each exchange is done in a disruption-tolerant manner, the overall application may perform poorly. SMTP, for example, is a relatively 'chatty' protocol that requires several round trips to accomplish its goal. Figure 4 shows the series of message exchanges when a standard (non-pipelined) SMTP client wants to send mail. The unmarked exchanges at the top and bottom of the timeline represent the setup and teardown of the TCP connection. If channel accesses take a long time, the number of channel accesses can have severe adverse affects on performance.

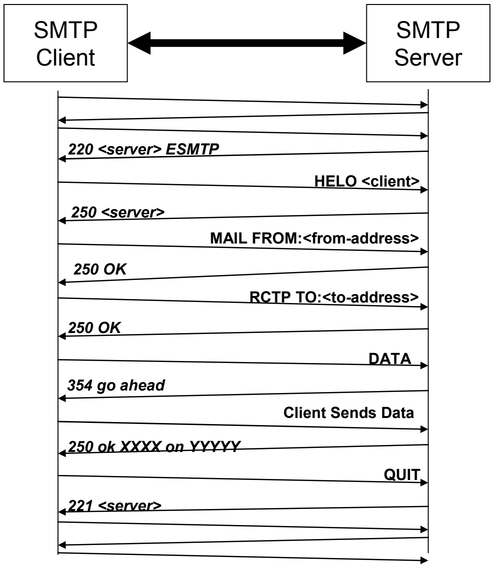


Figure 4: SMTP Interactions required to send an email message.

DTN alone can't do much to improve application performance in this case.

One solution to these problems is to redesign applications to be disruption tolerant and to recompile them to use the DTN API. While this is architecturally appealing, it requires custom modifications to each application, plus 'big-bang' application upgrades, including in those portions of the network with good connectivity.

A second option that allows augmenting existing applications with DTN technologies is to implement application-layer DTN proxies. Application-layer proxies terminate the protocol used by the application to exchange information, and translate that protocol into something else for transport over the disrupted portion of the network. Thus an application-layer proxy for SMTP would behave like an SMTP server (to SMTP clients), responding with the appropriate SMTP messages.

Such an SMTP proxy would have to generate the appropriate SMTP responses to cause the client to perform its task, such as transmitting its email messages. The proxy can then use some other mechanism, such as DTN, to ship information about the application-layer action to a peer proxy. The peer proxy receives the information and uses it to execute an SMTP exchange with a 'real' SMTP server, this time behaving as if it were the original SMTP client application.

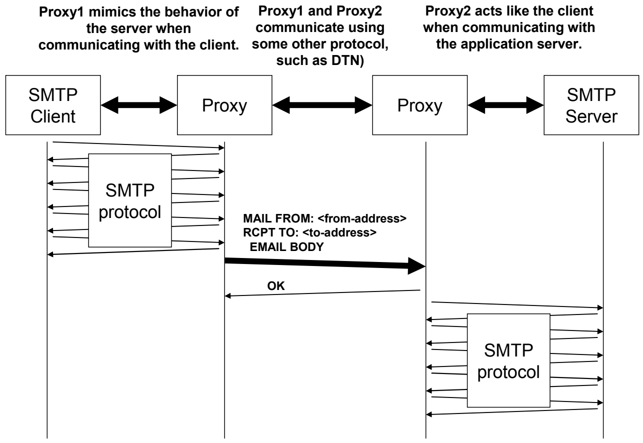

Proxy1 mimics the behavior of the server when communicating with the client.
Proxy1 and Proxy2 communicate using some other protocol, such as DTN)
Proxy2 acts like the client when communicating with the application server.
SMTP Client
Proxy
Proxy
SMTP Server
SMTP protocol
MAIL FROM: <from-address>
RCPT TO: <to-address>
EMAIL BODY
OK
SMTP protocol

Figure 5: An application-layer SMTP proxy.

Proxies of this type can interfere with application behavior. If the proxy cannot 'stall' the client until it is sure that the message has been delivered to the destination, then the client may believe that data has been delivered when in fact it is merely sitting in the proxy waiting to be forwarded. This is the case shown in Figure 5, where the proxy executes the entire SMTP protocol with the client (including the shutdown) before forwarding the information via the inter-proxy protocol. In most cases this is not a problem, but it is a drawback of the proxy approach.

A DTN-ENABLED WEB PROXY

This section describes an application layer DTN-http (web) proxy. We chose http partly because of the many web services that have battlefield applications [7], and partly because of the plethora of offline web browser utilities with source code available.

We implemented the DTN/http proxy by extending the World Wide Web Offline Explorer (WWWOFFLE) [8] to use DTN messages between a client and a server side. WWWOFFLE runs on Linux and was designed to be a web cache/offline viewer for users with a dialup or other intermittent connection to the Internet, as shown in Figure 6.

In its standard configuration, a single WWWOFFLE cache is placed between an isolated network and the Internet. The cache receives http requests from standard web browsers on the private network and, if not connected to the Internet, stores them in the local filesystem for later retrieval. When connected to the Internet, WWWOFFLE reads all of the requests in its outgoing directory, fetches all of the requested pages, and stores the responses in its web cache.

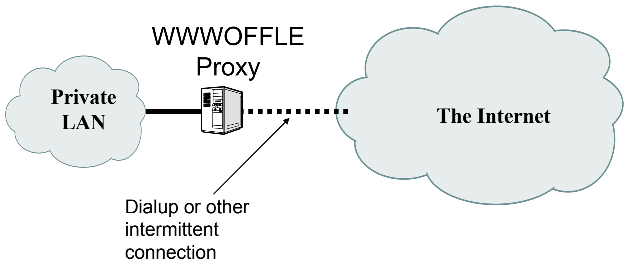
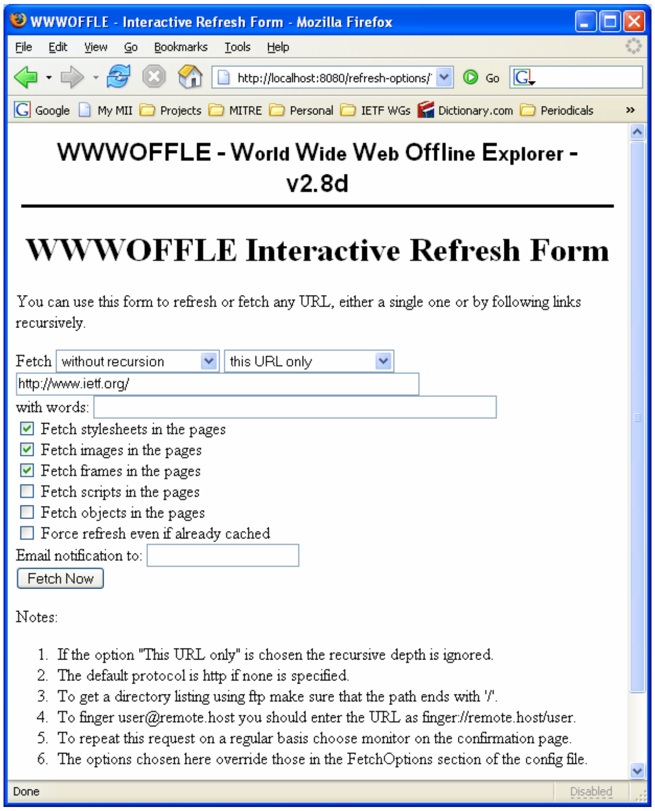


Figure 6: Typical WWWOFFLE configuration.

WWWOFFLE was attractive because it stores all of its state information (http requests, cached pages) in the file-system. This facilitated the collection and shipment of the state via DTN bundles. We investigated other web caches, including the popular Squid cache [9], but Squid is more complex and keeps much of its state in memory. Also, while the Internet Cache Protocol (ICP) that Squid uses might conserve bandwidth by allowing a tactical user to request information from the closest Squid peer, it generally requires at least two round trips to retrieve the data (one for the ICP query/response, and a second to retrieve the data). If the user is separated from the data source by a disrupted network with possibly large latency, the added round trip imposed by ICP could be cumbersome. Additionally, for the Jump-C2 application we're investigating, most requests will be filled by a well-known peer, obviating the need for ICP.

Conceptually, we split the WWWOFFLE proxy down the middle and form a client and a server side. The client side lives on the challenged or tactical piece of the network, and uses DTN bundles to communicate with the server side. The server side is assumed to be well-connected, so that when it receives requests from clients, it can use http on its well-connected side to retrieve the requested web pages.

The client side proxy receives http requests as usual, including all of WWWOFFLE's support for scoped recursion from the requested page, suppressing images, stylesheets, and frames. We implemented an additional notification field that allows the user to specify the target of an IM or email notification once the requested page(s) are loaded into the local cache. Figure 7 shows the form presented to the user when a page is not in the cache.


WWWOFFLE - Interactive Refresh Form - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:8080/refresh-options/
Go
Google My MII Projects MITRE Personal IETF WGs Dictionary.com Periodicals

WWWOFFLE - World Wide Web Offline Explorer - v2.8d

WWWOFFLE Interactive Refresh Form

You can use this form to refresh or fetch any URL, either a single one or by following links recursively.

Fetch without recursion this URL only
http://www.ietf.org/
with words:
☑ Fetch stylesheets in the pages
☑ Fetch images in the pages
☑ Fetch frames in the pages
☐ Fetch scripts in the pages
☐ Fetch objects in the pages
☐ Force refresh even if already cached
Email notification to:
Fetch Now

Notes:

1. If the option "This URL only" is chosen the recursive depth is ignored.
2. The default protocol is http if none is specified.
3. To get a directory listing using ftp make sure that the path ends with '/'.
4. To finger user@remote.host you should enter the URL as finger://remote.host/user.
5. To repeat this request on a regular basis choose monitor on the confirmation page.
6. The options chosen here override those in the FetchOptions section of the config file.

Figure 7: The WWWOFFLE refresh page.

Once the requests are written into the filesystem, they are tar'ed and compressed, and the resulting file is sent as a DTN bundle to the server-side WWWOFFLE proxy. The requests are then removed from the client filesystem so that they are not processed again later. A DTN-enabled thread in the server proxy receives bundled requests, uncompresses them into the correct directory, and then kicks the main proxy code to process the requests. Once the results are collected, pages that do not match the keyword search criteria are removed, and the remaining results are compressed and sent back to the client as a DTN bundle. The cache directory on the server side is then cleared to prevent the results from being resent in response to a subsequent request.

A DTN-aware thread in the client proxy receives the DTN bundle containing the compressed web pages from the server and extracts them to the WWWOFFLE cache directory. The client processes the received pages and forms a report with links to the received pages in case they are not all accessible from the top-level request. This can occur, for example, if the user requests recursion and search terms, and the graph of pages returned is not fully connected. This report is placed in the client proxy's local cache. Finally, if the user requested notification on receipt of the pages, an email or IM message is generated containing the requested URL and a pointer to the report. The process is illustrated in Figure 8.

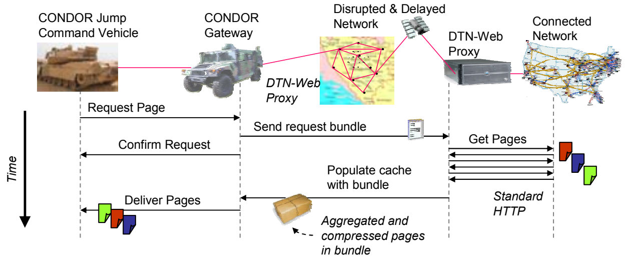


Figure 8: A modified WWWOFFLE web proxy enhanced with DTN functionality.

The server side DTN/WWWOFFLE proxy may support multiple clients, and requests may arrive asynchronously. To keep from either erroneously truncating user's requests or mixing the results of several requests, the server side proxy can be configured to use separate directories to process each client request.

Extending the WWWOFFLE proxy to use DTN bundles required under 500 lines of code, which includes filtering the results to include only those that contain the requested keywords, building the report pages, and notifying the user that new pages have arrived. This does not include the core DTN code, which is itself about 33,000 lines.

CONCLUSIONS AND FUTURE WORK

Network Centric Warfare is going to need support for disconnected / disrupted operations. While research into hardening the physical and data links to improve connectivity is certainly worthwhile, it appears that there will continue to be times when end-to-end connectivity cannot be assured. Thus IP implementations' reliance on end-to-end paths makes them unsuitable for use in tactical networks. By providing a networking technology that can efficiently traverse highly disrupted portions of the network and 'wait it out' when the network becomes disconnected, DTN can defend applications from the vagaries of intermittent network connectivity.

REFERENCES

[1] "Network Centric Warfare", Department of Defense report to congress, http://www.dod.mil/nii/NCW/

[2] W. Brown, V. Marano IV, W. MacCorkell and T. Krout, 'Future combat system-scalable mobile network demonstration performance and validation results' MILCOM 2003, 1286-1291.

[3] IRTF Delay Tolerant Research Group website - http://www.dtnrg.org

[4] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb and A. Joglekar, 'An Integrated Experimental Environment for Distributed Systems and Networks,' Proc. of the 5th Symposium on Operating Systems Design and Implementation, Boston MA, 2002

[5] M. Demmer, E. Brewer, K. Fall, S. Jain, M. Ho and R. Patra, " Implementing Delay Tolerant Networking," IRB-TR-04-020, Dec. 28, 2004, http://www.dtnrg.org/papers/demmer-irb-tr-04-020.pdf

[6] D. Mohney, "Soaring Condor Relieves Headaches," Mobile Radio Technology, June 2004, http://www.findarticles.com/p/articles/mi_m0HEP/is_6_22/ai_n6067603

[7] Lightweight Collaborative Whiteboard, http://www.mitre.org/news/events/tech04/briefings/1474.pdf

[8] http://www.gedanken.demon.co.uk/wwwoffle/

[9] D. Wessels and K. Claffy, ' ICP and the Squid web cache,' IEEE Journal on Selected Areas in Communications, Volume 16, Issue 3, April 1998 Page(s): 345 - 357