with intermittent (predictable and unknown) connectivity, large and/or variable delays, and high bit error rates. To provide its services over existing domain specific protocols, the DTN protocols reside at the application layer of the TCP/IP stack, forming a store-and-forward overlay network. The key capabilities of the Bundle Protocol include custody-based reliability, the ability to cope with intermittent connectivity, the ability to take advantage of scheduled and opportunistic connectivity, and late binding of names to addresses.

Internet standards are published in Request For Comments (RFCs), and the Bundle Protocol and LTP are described in RFC 5050 and RFC 5326, respectively. BP provides the store-carry-forward, custody transfer and naming capabilities of the DTN, while LTP was specifically developed for long-delay links. LTP allows for "red" and "green" data portions in a single session, where the red data portion uses retransmission and the green data portion does not. Unlike common Internet retransmission protocols, LTP adds the ability to suspend and resume timers when the link's status changes. On occasion, the models are extended to include non-standard experimental features for validating project-specific performance or behavioral requirements. For instance, unlike standard simulation models, the BP model supports external traffic injection, which was used to verify correct behavior of the SharedNet middleware over DTN protocols and described at the SMC-IT 2006 conference (Second International Conference On Space Mission Challenges For Information Technology). The MACHETE LTP model supports all standard functions of LTP along with an optional priority-aware queuing system to prevent lower priority from blocking higher-priority traffic arriving later.

Furthermore, MACHETE contains Consultative Committee for Space Data Systems (CCSDS) protocol standards, such as Proximity-1, Advanced Orbiting Systems (AOS), Packet Telemetry/Telecommand, Space Communications Protocol Specification (SCPS), and the CCSDS File Delivery Protocol (CFDP). So, with the addition of DTN protocol libraries interplanetary network, engineers at JPL can characterize future space network performance trade-offs.

This work was done by John S. Seguí, Esther H. Jennings, and Jay L. Gao of Caltech for NASA's Jet Propulsion Laboratory. For more information, contact iaoffice@jpl.nasa.gov.

This software is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-43410.

Contact Graph Routing

Contact Graph Routing (CGR) is a dynamic routing system that computes routes through a time-varying topology of scheduled communication contacts in a network based on the DTN (Delay-Tolerant Networking) architecture. It is designed to enable dynamic selection of data transmission routes in a space network based on DTN. This dynamic responsiveness in route computation should be significantly more effective and less expensive than static routing, increasing total data return while at the same time reducing mission operations cost and risk.

The basic strategy of CGR is to take advantage of the fact that, since flight mission communication operations are planned in detail, the communication routes between any pair of "bundle agents" in a population of nodes that have all been informed of one another's plans can be inferred from those plans rather than discovered via dialogue (which is impractical over long one-way-light-time space links). Messages that convey this planning information are used to construct "contact graphs" (time-varying models of network connectivity) from which CGR automatically computes efficient routes for bundles. Automatic route selection increases the flexibility and resilience of the space network, simplifying cross-support and reducing mission management costs.

Note that there are no "routing tables" in Contact Graph Routing. The best route for a bundle destined for a given node may routinely be different from the best route for a different bundle destined for the same node, depending on bundle priority, bundle expiration time, and changes in the current lengths of transmission queues for neighboring nodes; routes must be computed individually for each bundle, from the Bundle Protocol agent's current network connectivity model for the bundle's destination node (the contact graph). Clearly this places a premium on optimizing the implementation of the route computation algorithm. The scalability of CGR to very large networks remains a research topic.

The information carried by CGR contact plan messages is useful not only for dynamic route computation, but also for the implementation of rate control, congestion forecasting, transmission episode initiation and termination, timeout interval computation, and retransmission timer suspension and resumption.

This work was done by Scott C. Burleigh of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

This software is available for commercial licensing. Please contact Daniel Broderick of the California Institute of Technology at danielb@caltech.edu. Refer to NPO-45488.

Parallel Eclipse Project Checkout

Parallel Eclipse Project Checkout (PEPC) is a program written to leverage parallelism and to automate the checkout process of plug-ins created in Eclipse RCP (Rich Client Platform). Eclipse plug-ins can be aggregated in a "feature project." This innovation digests a feature description (xml file) and automatically checks out all of the plug-ins listed in the feature. This resolves the issue of manually checking out each plug-in required to work on the project. To minimize the amount of time necessary to checkout the plug-ins, this program makes the plug-in checkouts parallel. After parsing the feature, a request to checkout for each plug-in in the feature has been inserted. These requests are handled by a thread pool with a configurable number of threads. By checking out the plug-ins in parallel, the checkout process is streamlined before getting started on the project.

For instance, projects that took 30 minutes to checkout now take less than 5 minutes. The effect is especially clear on a Mac, which has a network monitor displaying the bandwidth use. When running the client from a developer's home, the checkout process now saturates the bandwidth in order to get all the plug-ins checked out as fast as possible. For comparison, a checkout process that ranged from 8-200 Kbps from a developer's home is now able to saturate a pipe of 1.3 Mbps, resulting in significantly faster checkouts.

Eclipse IDE (integrated development environment) tries to build a project as soon as it is downloaded. As part of another optimization, this innovation programmatically tells Eclipse to stop building while checkouts are happening, which dramatically reduces lock contention and enables plug-ins to continue downloading until all of them finish. Furthermore, the software re-enables