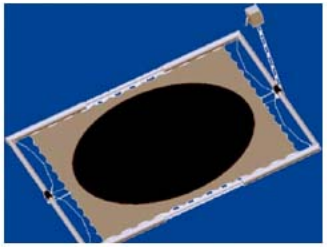
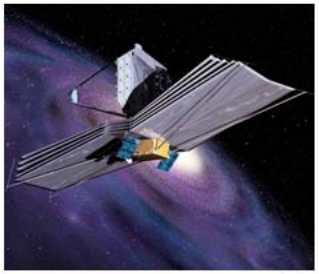
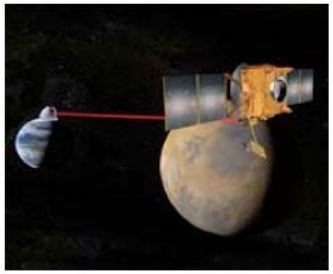
Jet Propulsion Laboratory
California Institute of Technology

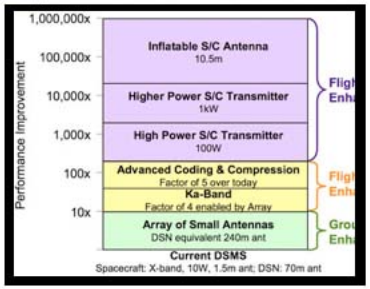
SpaceOps 2008

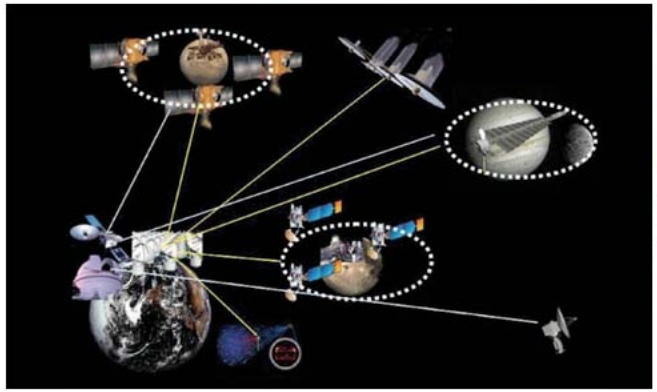
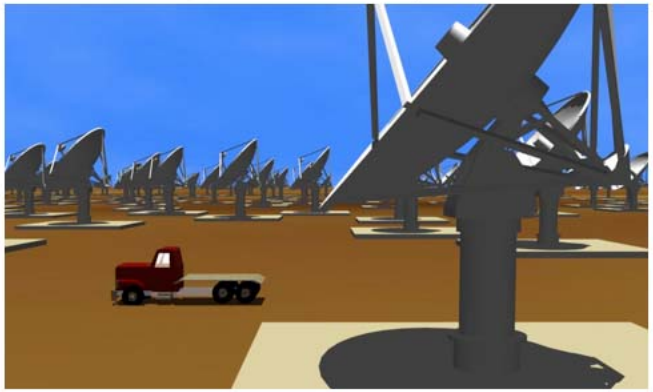


Dynamic Routing for Delay-Tolerant Networking in Space Flight Operations

Scott Burleigh
Jet Propulsion Laboratory, California Institute of Technology



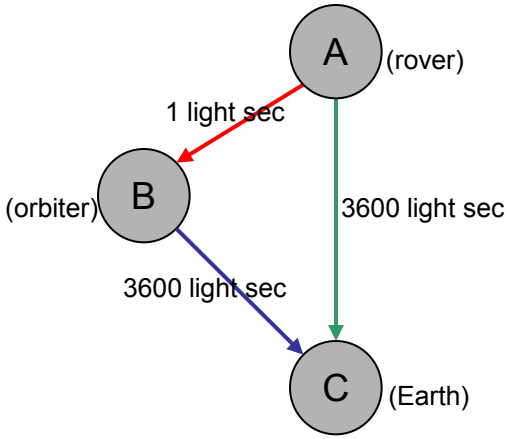



Contact Graph Routing
Motivation

- Large-scale future space exploration will offer complex communication challenges that may be best addressed by establishing a network infrastructure.
- The Internet protocols are not well suited for operation of a network over interplanetary distances; a *Delay-Tolerant Networking* (DTN) architecture has been proposed instead.
- Traffic flow within a delay-tolerant network, as in the Internet, will be more efficient if routers can automatically select different forwarding paths at different times, depending on nodes' anticipated ability to forward data on a timely basis.
- However, Internet techniques for route computation are, again, not well suited for operating a network over interplanetary distances where changes in topology may occur more rapidly than they can be reported.
- *Contact Graph Routing* (CGR) offers an alternative, delay-tolerant technique for dynamic route computation in a delay-tolerant network.

Contact Graph Routing
An Example Scenario

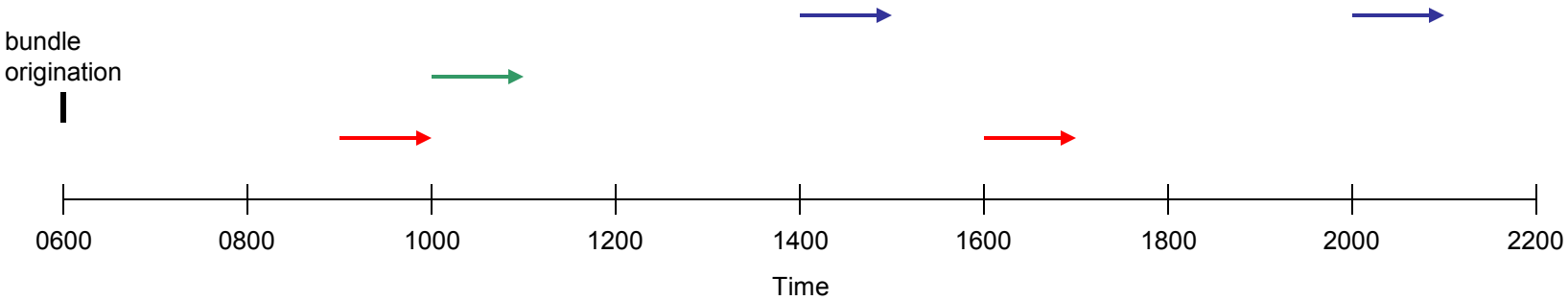
50,000-byte bundle originated at A at 0600, destined for C.



Each A → C contact is 1 hr at 8000 bps, total capacity 3,600,000 bytes.

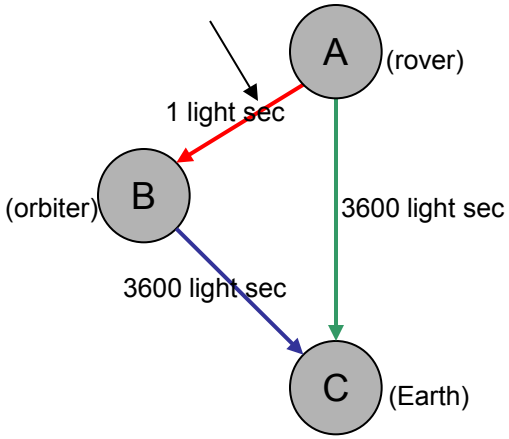
Each A → B contact is 1 hr at 80,000 bps, total capacity 36,000,000 bytes.

Each B → C contact is 1 hr at 20,000 bps, total capacity 9,000,000 bytes.



Contact Graph Routing
Case 1: Bundle TTL is 12 hours

50,000-byte bundle originated at A at 0600, destined for C.



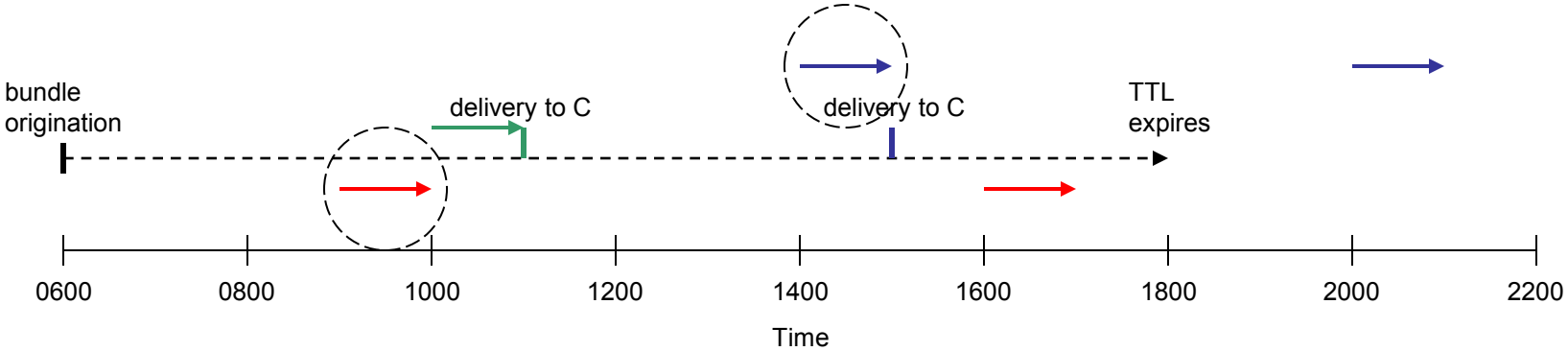
Each A → C contact is 1 hr at 8000 bps, total capacity 3,600,000 bytes.

Each A → B contact is 1 hr at 80,000 bps, total capacity 36,000,000 bytes.

Each B → C contact is 1 hr at 20,000 bps, total capacity 9,000,000 bytes.

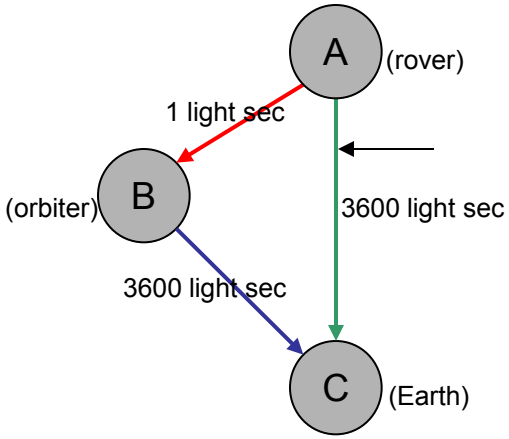
1. Either the A→C contact or the first B→C contact will deliver the bundle prior to TTL expiration.
2. The first A→B contact will deliver the bundle prior to the end of the first B→C contact.
3. So either A→C or A→B→C can succeed, so sending directly to either B or C can succeed.
4. The *bottleneck* (lowest-capacity link) on the A→B→C path is 9 million bytes. The bottleneck on the A→C path is 3.6 million bytes.
5. Since the A→B→C path is less constricted, queue the bundle for transmission to B rather than directly to C.

Note that this leaves the A→C contact available for other bundles that may need it, and it maximizes utilization of the high-capacity links.



Contact Graph Routing
Case 2: Bundle TTL is 7 hours

50,000-byte bundle originated at A at 0600, destined for C.

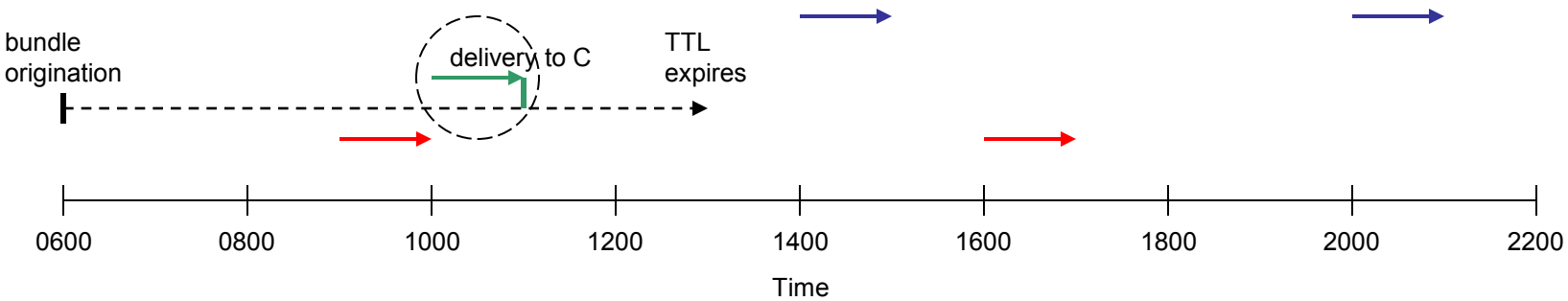


Each A → C contact is 1 hr at 8000 bps, total capacity 3,600,000 bytes.

Each A → B contact is 1 hr at 80,000 bps, total capacity 36,000,000 bytes.

Each B → C contact is 1 hr at 20,000 bps, total capacity 9,000,000 bytes.

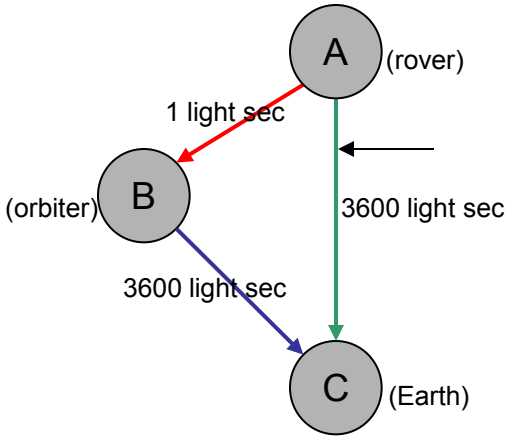
1. Only the A→C contact can deliver the bundle prior to TTL expiration.
2. So we must queue the bundle for transmission directly to C.



Contact Graph Routing

Case 3: TTL is 12 hours, but 36 MB queued for B

50,000-byte bundle originated at A at 0600, destined for C.

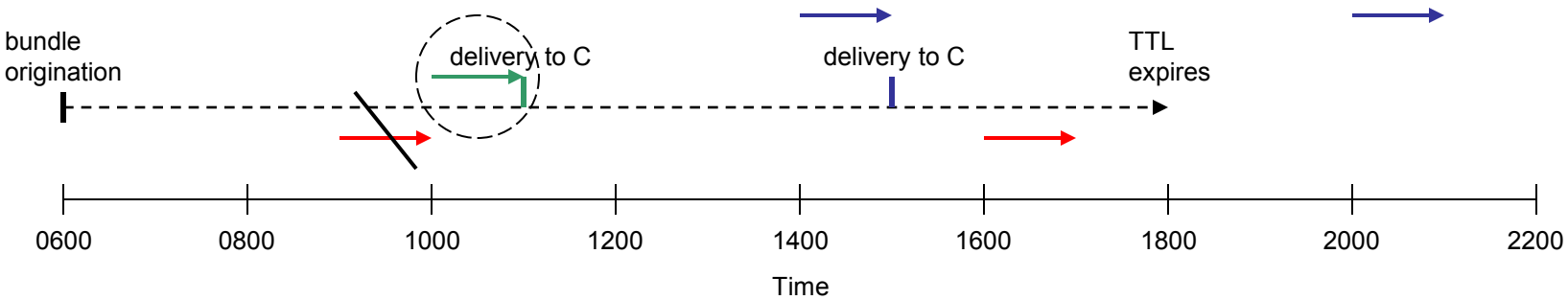


Each A → C contact is 1 hr at 8000 bps, total capacity 3,600,000 bytes.

Each A → B contact is 1 hr at 80,000 bps, total capacity 36,000,000 bytes.

Each B → C contact is 1 hr at 20,000 bps, total capacity 9,000,000 bytes.

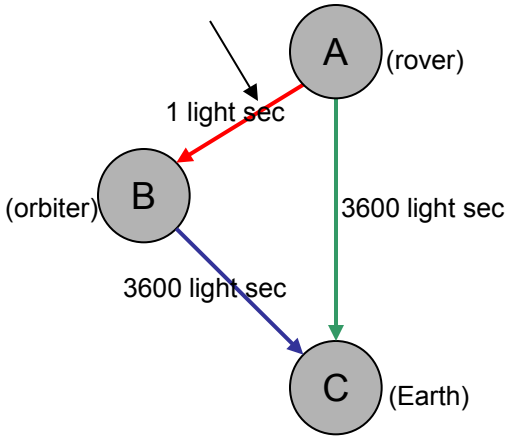
1. Either the A→C contact or the first B→C contact will deliver the bundle prior to TTL expiration.
2. The first A→B contact will deliver the bundle prior to the start of the first B→C contact, but 36 million bytes are already queued on it.
3. The second A→B contact will not deliver the bundle prior to the end of the first B→C contact, so routing via B will not succeed.
4. So we can only queue the bundle for transmission directly to C.



Contact Graph Routing

Case 4: TTL is 16 hours, with 36 MB queued for B

50,000-byte bundle originated at A at 0600, destined for C.

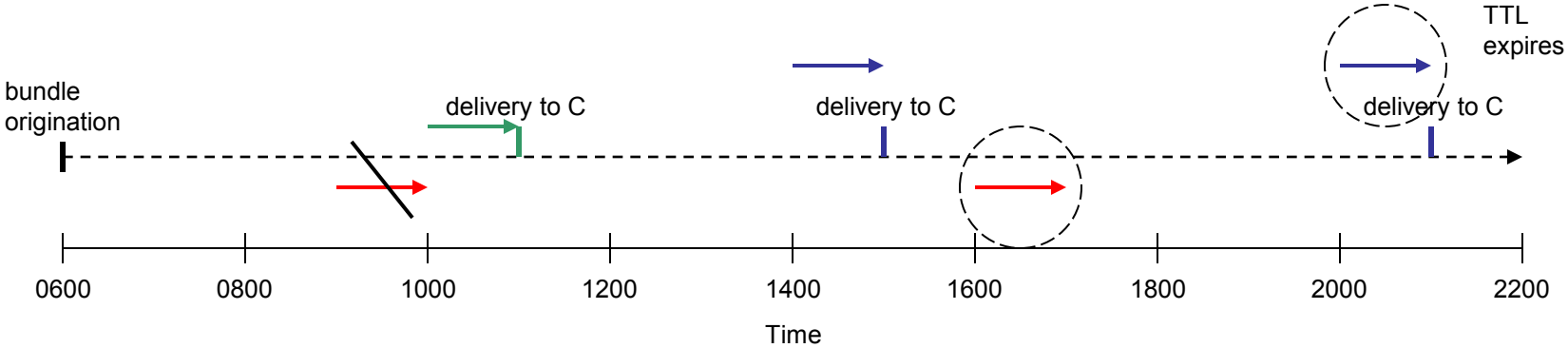


Each A → C contact is 1 hr at 8000 bps, total capacity 3,600,000 bytes.

Each A → B contact is 1 hr at 80,000 bps, total capacity 36,000,000 bytes.

Each B → C contact is 1 hr at 20,000 bps, total capacity 9,000,000 bytes.

1. Either the A→C contact or the first <u>or second</u> B→C contact will deliver the bundle prior to TTL expiration.
2. The first A→B contact will deliver the bundle prior to the start of the first B→C contact, but 36 million bytes are already queued on it.
3. The second A→B contact will deliver the bundle prior to the end of the second B→C contact.
4. So again either A→C or A→B→C can succeed, so sending directly to either B or C can succeed.
5. The *bottleneck* (lowest-capacity link) on the A→B→C path is 9 million bytes. The bottleneck on the A→C path is 3.6 million bytes.
6. Since the A→B→C path is less constricted, queue the bundle for transmission to B rather than directly to C.



Contact Graph Routing
Exception Handling (1): Contact Failure

- We queue a bundle for transmission to a node in the expectation that it will be transmitted during an identified contact.
- If that contact happens to be canceled or truncated, or data rate is less than expected, then that transmission may not occur.
- So we set a timer for the end of the identified contact for each queued bundle. If the timer expires before the bundle has been transmitted, we re-compute the route for this bundle and re-queue it for transmission, possibly to a different node.
  - Leaving it in place in its current queue could make other bundles in the queue miss <u>their</u> transmission deadlines, with potential for widespread delivery failure.

Contact Graph Routing
Exception Handling (2): Custody Refusal

- We queue a bundle for transmission to a node in the expectation that this node can receive, store, and successfully forward the bundle toward its final destination.
- If the receiving node happens to have insufficient storage or to be unable to compute a forward route for the bundle, then reception may fail. If the bundle is non-custodial, it is simply dropped; if it is custodial, a custody refusal signal is returned to the sending node.
- On receiving a custody refusal signal:
  - We re-compute the route for this bundle and for all others that are currently queued for transmission to the refusing node, re-queuing them for transmission to a different node.
  - We remember this refusal. When computing routes for future bundles with the same final destination we ignore otherwise plausible routes that would entail transmission to the refusing node.
  - Periodically we send a “probe” bundle to the refusing node to see if it is still refusing this traffic. If custody is accepted, we forget this refusal when computing routes for future bundles.

Contact Graph Routing
Conclusion

- Computational self-sufficiency – the making of communication decisions on the basis of locally available information that is already in place, rather than on the basis of information residing at other entities – is a fundamental principle of Delay-Tolerant Networking.
- Contact Graph Routing is an attempt to apply this principle to the problem of dynamic routing in an interplanetary DTN.
- Testing continues, but preliminary results are promising.