TCP Connection Migration

Alex C. Snoeren
and Hari Balakrishnan

MIT Lab for Computer Science

June 26, 2000

`http://nms.lcs.mit.edu/projects/migrate`

Why do Connection Migration?

- Mobility
  - End hosts may change network attachment points
- Load Balancing / Fault Tolerance
  - Servers may want to off-load long-lived flows
- Redirection reduces to Migration
- Anycasting
  - Rebind a connection to a unicast address

Previous Approaches

- Connection-ID schemes
  - Constant overhead on *every* packet
  - Lots of security and DoS issues
- Mobility Schemes
  - Mobile-IP
    - Completely transparent to end hosts
    - Requires a *home agent*
    - Introduces routing anomalies (triangle routing)
  - EID schemes
    - Yet another level of indirection

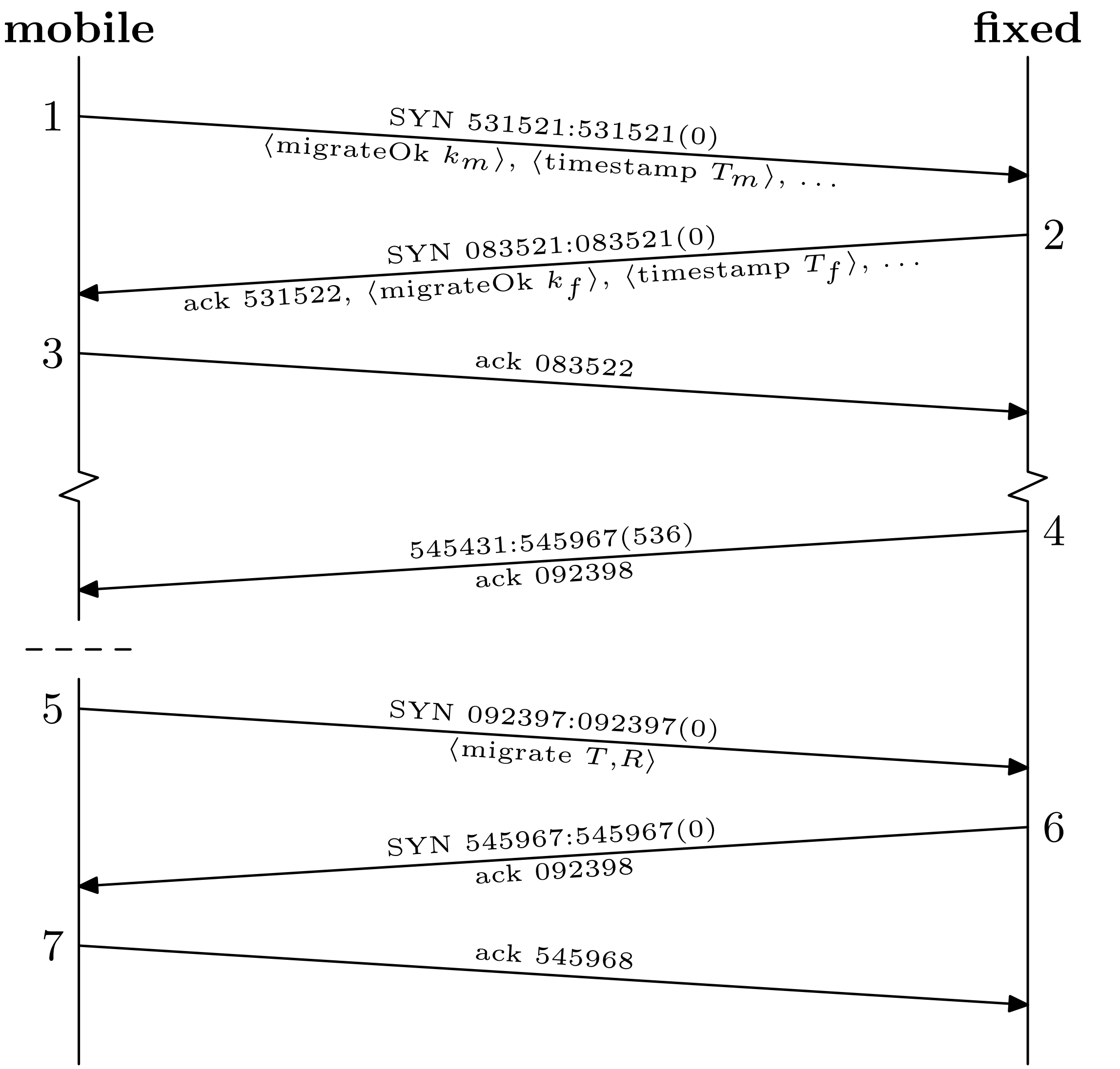
Our Approach

- Migrate TCBs from established connections
  - Special SYN packets include a *Migrate* option
    - Migrate SYNs do not establish new connections, but migrate previously-established ones
    - Established connections are referenced by a *token*
  - Maintain all old state (sequence space, options, etc.)
  - Tokens negotiated during connection establishment
- No need to anticipate re-addressing
  - New (continuing) connections can be from anywhere
  - Special RST handling for address reassignment

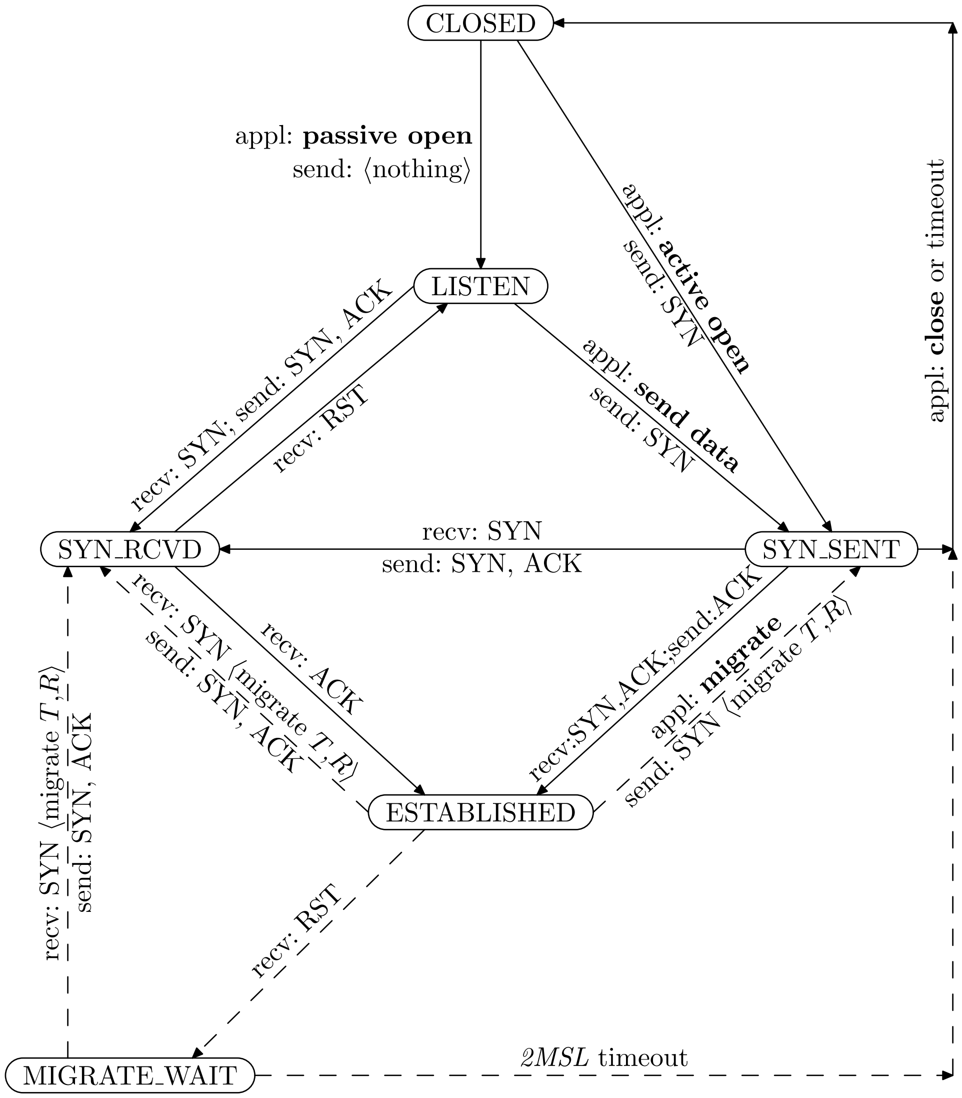
Benefits

- No per-packet overhead
- Minimal changes to TCP state machine
- No changes to TCP semantics
  - An established connection is migrateable
  - End-host TCP options work
    - SACK, FACK, Timestamps, etc.
  - Network extensions work
    - Snoop, NATs, etc.

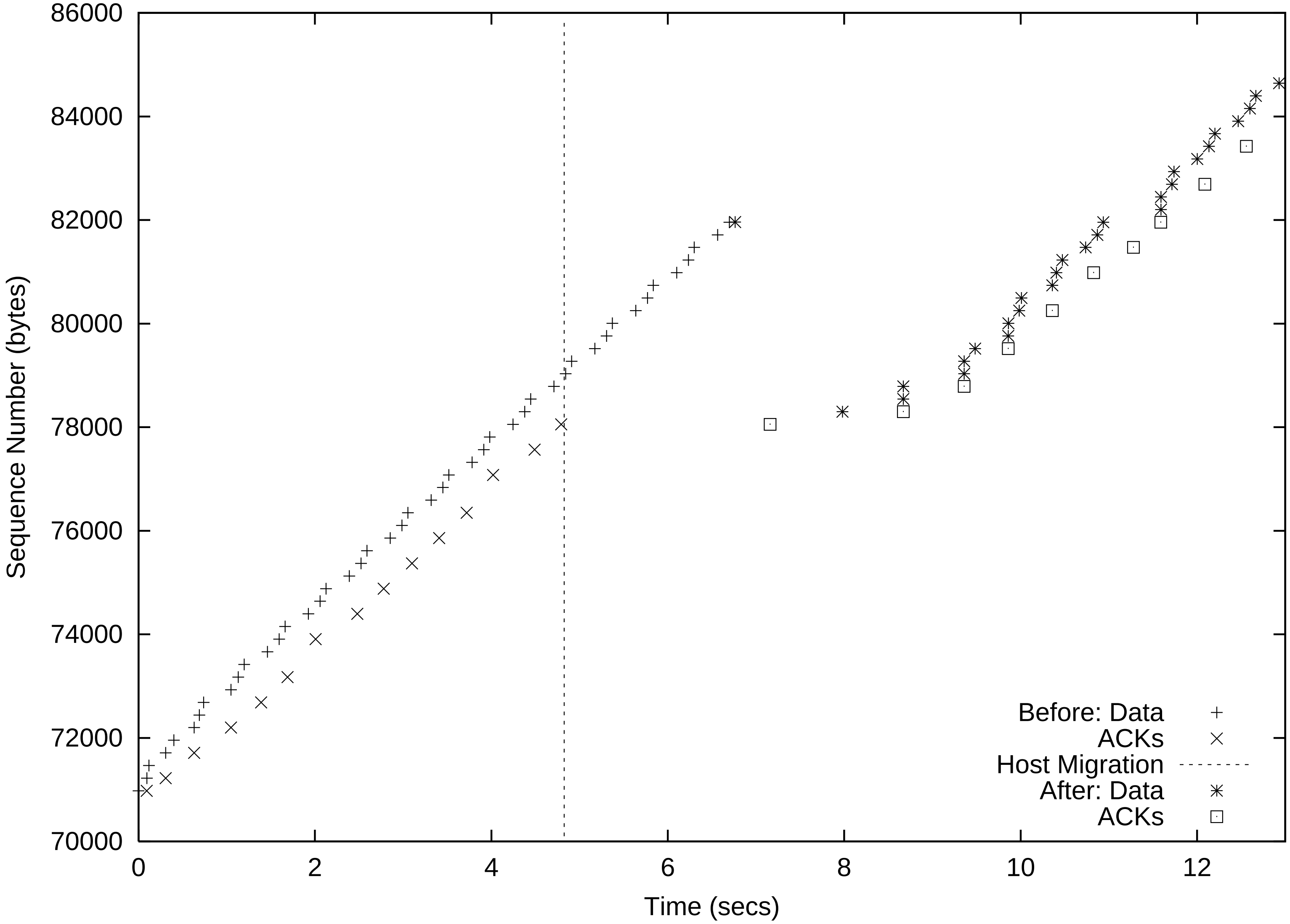
TCP Connection Migration



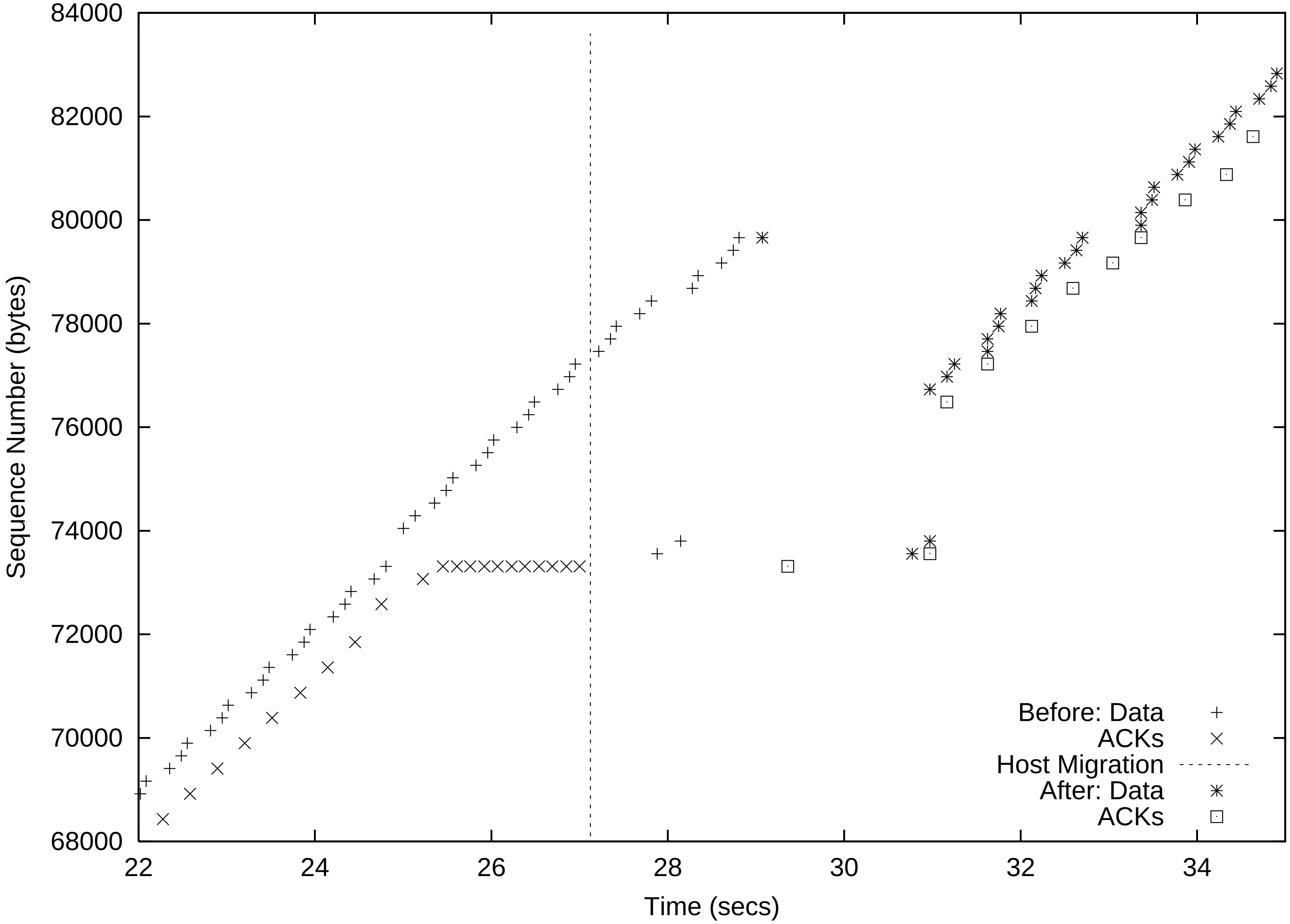
TCP State Diagram Modifications



A Migrate Connection Trace



A SACK Trace with Losses



Securing the Migration

- Migrate requests are as secure as the sequence space
- No need for further security with IPsec
- Without IPsec, could be sniffed and spoofed
  - *The case for TCP today without ingress filtering!*
  - Ingress filtering doesn't help with Migrate requests
  - Optionally secure requests with a secret key, K
  - Negotiate the secret key in-band with ECDH
  - Requests have two parts to avoid DoS attacks
    - A *pre-computable* secret nonce
    - An unforgeable migration request

TCP Migrate option

	Kind: 16	Length = 19	ReqNo
Token			
Token (cont.)			
Request			
Request (cont.)			

$$Token = SHA1(N_i, N_j, K)$$

$$Request = SHA1(N_i, N_j, K, SeqNo, ReqNo)$$

TCP Migrate-Permitted option

<table>
<tr><td>Kind: 15</td><td>Length = 3/20</td><td>Curve Name</td><td>ECDH PK</td></tr>
<tr><td colspan="4">ECDH Public Key (cont.)</td></tr>
<tr><td colspan="4">ECDH Public Key (cont.)</td></tr>
<tr><td colspan="4">ECDH Public Key (cont.)</td></tr>
<tr><td colspan="4">ECDH Public Key (cont.)</td></tr>
</table>