DCCP Mobility and Multihoming

<draft-kohler-dccp-mobility-01.txt>

Draft: Eddie

Slides: Pasi & Eddie
Motivation

• Multi-homing is relevant use case for multi-access mobiles
  • Wireless Wide-Area link (e.g., GPRS) is usually available most of the time
  • Wireless LAN access can be short-lived
    • Usually mobile terminals can keep both interfaces up at the same time
  • Sometimes using Mobile IP is not possible
    • Requires support in network: the home agent, sometimes a foreign agent in IPv4
• Multi-homing on transport layer has nice characteristics
  • Requires support only at the end hosts
  • Supports hand-offs between IPv4 and IPv6
  • Multiple parallel paths per connection
    • Mobility decisions can be made independently for each flow
• Often the location of server is fixed and known
Basics

• New version introduced the idea of Generalized Connections (Gencon)
• DCCP connections that support multi-homing are assigned Gencon ID
• Component connections are initialized and maintained as separate connections
  • Separate congestion control state
  • Use DCCP-Request, DCCP-Response, and DCCP-Ack as with normal DCCP connections
  • Include Gencon option that contains a Gencon message
  • Connections under same Gencon are owned by same socket
• Socket is closed when the last connection under Gencon is closed
• Reconfiguration of address set is protected with encrypted random nonce
  • Crypto algorithms not yet specified
• Some requirements
  • Must be safe against hijacking
  • Must be able to move between different NAT domains
Gencon Message Types (Initialization)

• (All messages contain 64-bit Gencon ID and 32-bit Component ID)
  • Client defines upper 32 bits in Gencon ID, server defines lower 32 bits

• **Initiate**: with DCCP-Request
  • Includes key type and client’s public key

• **Approve**: with DCCP-Response
  • Server gives its public key
Gencon Message Types (Mobility)

- **Attach:** Assign a new address to Gencon (with DCCP-Request)
  - Includes new Component ID and an optional 64-bit random nonce to verify server
  - Component “client” can be different than the original Gencon client
- **Challenge:** To verify the identity of client (with DCCP-Response)
  - 64-bit random nonce is mandatory
  - Optional 40-byte encrypted token composed from: Message type, sequence number, acknowledgment number, Gencon ID, Component ID, and Server Nonce
- **Confirm:** To complete client identity verification (with DCCP-Ack)
  - Includes 40-byte encrypted token based on the above mentioned variables at client
- **Detach:** Removes component connection from Gencon
  - Includes 40-byte encrypted token
Comments

• Primary component ID should be defined
  • Now: last connection on which data was received
  • Could use some sort of preference scoring?
  • Or, most recent Attach indicates most preferred?
• On “Random nonce MUST NOT be reused on same Gencon ID”
  • Might lead to heavy bookkeeping on long connections
  • New text on generating apparently-random nonces coming up
• How to negotiate common key type between end hosts
  • Could client indicate the set of key types it supports?
• Allocation of component IDs
  • Receiver MUST check sender does not reuse Component ID
  • Easiest to allocate IDs incrementally
Next steps?

- To working group charter?