Peer-to-Peer Networks
15 Self-Organization

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg
Topology-Management

- T-Man: Fast Gossip-based Construction of Large-Scale Overlay Topologies Mark Jelasity Ozalp Babaoglu, 1994
Distributed Topology Construction
T-Man

\textbf{do} at a random time once in each consecutive interval of \( T \) time units
\begin{align*}
  p & \leftarrow \text{selectPeer}() \\
  \text{myDescriptor} & \leftarrow (\text{myAddress}, \text{myProfile}) \\
  \text{buffer} & \leftarrow \text{merge}(\text{view}, \{\text{myDescriptor}\}) \\
  \text{buffer} & \leftarrow \text{merge}(\text{buffer}, \text{rnd.view}) \\
  \text{send buffer to } p & \\
  \text{receive buffer}_p \text{ from } p & \\
  \text{buffer} & \leftarrow \text{merge}(\text{buffer}_p, \text{view}) \\
  \text{view} & \leftarrow \text{selectView}(\text{buffer}) \\
\end{align*}
(a) active thread

\textbf{do} forever
\begin{align*}
  \text{receive buffer}_q \text{ from } q & \\
  \text{myDescriptor} & \leftarrow (\text{myAddress}, \text{myProfile}) \\
  \text{buffer} & \leftarrow \text{merge}(\text{view}, \{\text{myDescriptor}\}) \\
  \text{buffer} & \leftarrow \text{merge}(\text{buffer}, \text{rnd.view}) \\
  \text{send buffer to } q & \\
  \text{buffer} & \leftarrow \text{merge}(\text{buffer}_q, \text{view}) \\
  \text{view} & \leftarrow \text{selectView}(\text{buffer}) \\
\end{align*}
(b) passive thread

\textbf{Fig. 1.} The T-Man protocol.
Finding a Torus

Fig. 2. Illustrative example of constructing a torus over $50 \times 50 = 2500$ nodes, starting from a uniform random topology with $c = 20$. For clarity, only the nearest 4 neighbors (out of 20) of each node are displayed.
Convergence of T-MAN

\[ L^4 \cdot L^6 = 16,384 \]
T-Chord

- Chord on demand, A Montresor, M Jelasity, O Babaoglu - Peer-to-Peer Computing, 2005
- Apply self-organization to Chord
  - compare insertion operation Pastry
- T-Chord
  - Apply T-Man
  - preferring Chord edges
- T-Chord-Prox
  - rank according to RTT
Ranking Function T-Chord

- 1st rank
  - nearest successor/predecessor on the ring \([0, 2^m - 1]\)
- For each exponent \(j \in [1, m - 1]\)
  - select from view the nodes nearest to 
    \([ID + 2^j \mod 2^m, ID + 2^{j+1} - 1 \mod 2^m]\)
Ranking Function T-Chord-Prox

- 1st rank
  - nearest successor/predecessor on the ring \([0, 2^m - 1]\)
- For each exponent \(j \in [1, m - 1]\)
  - select from view the nodes nearest to \([\text{ID} + 2^j \mod 2^m, \text{ID} + 2^{j+1} - 1 \mod 2^m]\)
  - measure latency (RTT) for \(p\) random nodes from view in such intervals and choose the closest
Adaption for Chord

T-Man for T-Chord

- **selectPeer()**: randomly select a peer \( q \) from the \( r \) nodes in my view that are nearest to \( p \) in terms of ID distance
- **extract()**: send to \( q \) the \( r \) nodes in local view that are nearest to \( q \)
  - \( q \) responds with the \( r \) nodes in its view that are nearest to \( p \)
- **merge()**: both \( p \) and \( q \) merge the received nodes to their view

© 2005 Alberto Montresor
After Exchange of Links

T-Man for T-Chord

- `selectPeer()`: randomly select a peer `q` from the `r` nodes in my view that are nearest to `p` in terms of ID distance.

- `extract()`: send to `q` the `r` nodes in local view that are nearest to `q`.
  - `q` responds with the `r` nodes in its view that are nearest to `p`.

- `merge()`: both `p` and `q` merge the received nodes to their view.

© 2005 Alberto Montresor
T-Chord Performance

- Starting with a neighbors on the ring
- Loss rate and hop count
  - experiments on a real-word dataset from 2002
T-Chord Performance

- Starting with a neighbors on the ring
- Loss rate and hop count
T-Chord Performance

- Message Delay

![Graph showing Message Delay vs Size for Chord, T-Chord, and T-Chord Prox]
T-Chord Performance

- Robustness
Peer-to-Peer Networks
15 Self-Organization

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg