Simulation and Performance Analysis of Peer-to-Peer Network Games

ENSC 835 – High-Performance Networks

Final Project Presentation

Fall 2003

David Mikulec

Email: dmikulec@sfu.ca / Web: www.sfu.ca/~dmikulec

Roadmap

- Introduction to Network Games
- Defining the Challenge
- Implementation in ns-2
- Discussion of Results
- List of Remaining Work
- Conclusion

Introduction to Network Games

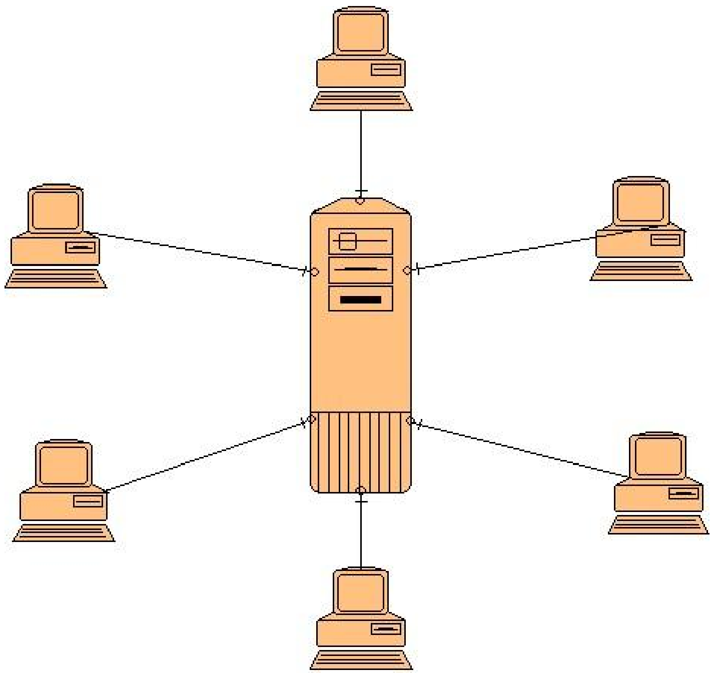
- What is a network game?
  - Two or more players on separate computers connected by LAN or Internet
  - One “world” whose state is maintained by network communication
  - Typically all network communication is application-layer

Introduction to Network Games

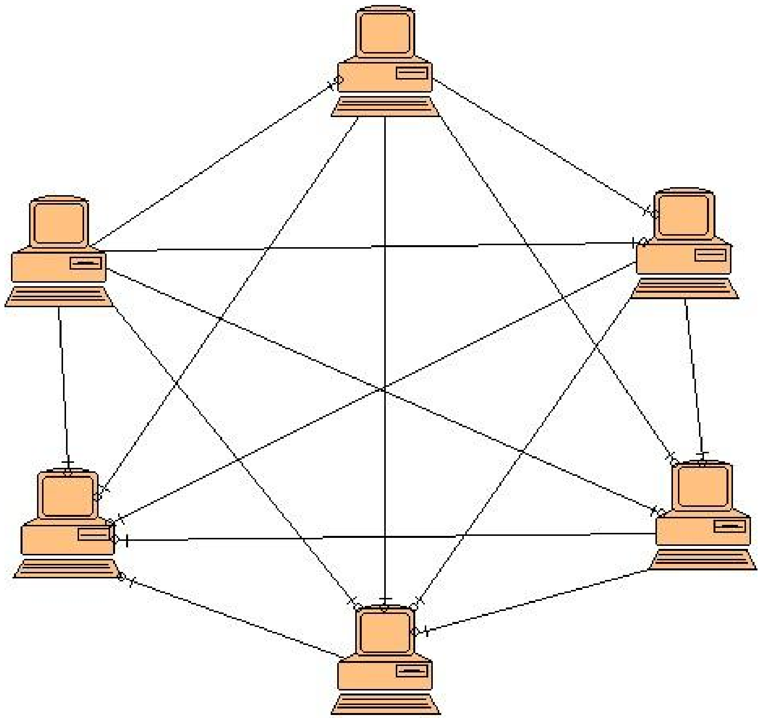
- Some examples
  - First Person Shooters (Quake)
  - Real Time Strategy (StarCraft)
  - Sports (NHL 2004)
  - Card / Board Games (Chess)
  - MMORPG (Everquest)

Introduction to Network Games

- Client-Server



- Peer to Peer



Introduction to Network Games

- Client-server advantages
  - Requires less bandwidth due to single connection
  - Lost packets affect only one client
- Peer-to-peer advantages
  - No server means no central point of failure
  - Lower delay due to direct connections

Introduction to Network Games

- Focus on Peer-to-peer networks because...
  - High bandwidth connections are becoming commonplace, currently ~1Mbps but future may bring orders of magnitude more
  - Higher round trip time (RTT) in client-server connections is here to stay (speed of light places lower bound on delay)

Defining the Challenge

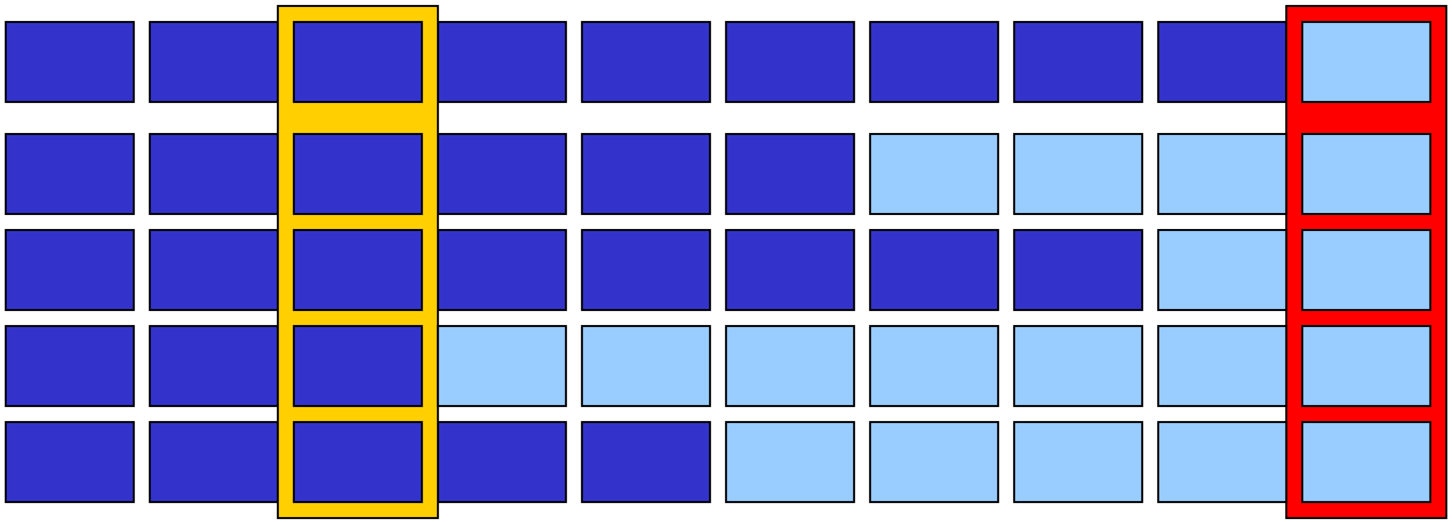
- Major challenge
  - Distributed game simulation must run identically on all machines – identical inputs required at each frame to guarantee synchronization. How can we achieve 30 frames per second (or more) given typical network delays of 20-100ms across North America (and even longer globally)?

Defining the Challenge

- The solution – apply inputs to the simulation engine several frames after their generation.
  - For example, wait 6 frames (or 200ms assuming 30 fps) – if all peers can consistently deliver data within 200ms the game will run smoothly
  - Any delay beyond the maximum causes the game to freeze while starved of data

Defining the Challenge

- For example, dark blue indicates packets received, light blue represent packets to be received. The top row is the local user's input, other rows represent remote peers. In this case, the yellow frame are the last full set of inputs, which will let us simulate up to, but not including the red frame.



Implementation in ns-2

- Simulate and analyze the performance of various configurations of fully-meshed peer-to-peer network games, variable parameters include:
  - Client connectivity (ADSL, modem, etc.)
  - Number of peers
  - Traffic characteristics (fixed / variable size)

Implementation in ns-2

- Key implementation decision – which transport layer protocol to use?
  - Peer-to-peer games require reliable communication which implies TCP
  - Real games however often choose UDP, why?
    - Save per-packet overhead by implementing a lighter-weight reliable application level protocol
    - Finer level of control over retransmissions, etc.

Implementation in ns-2

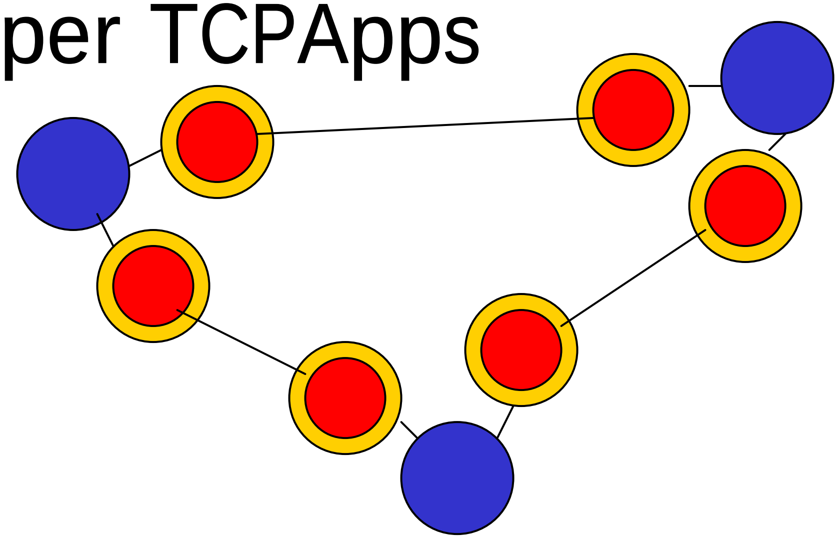
- TCP is the protocol of choice for this project because:
  - Implementing an extremely detailed application layer protocol which performs a similar function to TCP is time consuming
  - Interesting to view the performance of TCP as an alternative to complicated UDP solutions

Implementation in ns-2

- High-level simulation – use OTcl
- Built-in OTcl class TcpApp in ns-2 provides the necessary functionality to pass data between applications connected by TCP
- New GameApp class was defined to hold onto TcpApp connections to each peer, GameApp sends packets at a constant rate as long as no remote peer is more than 6 frames behind
- Modify labels and colours on nodes to identify when game is running smoothly or stuck

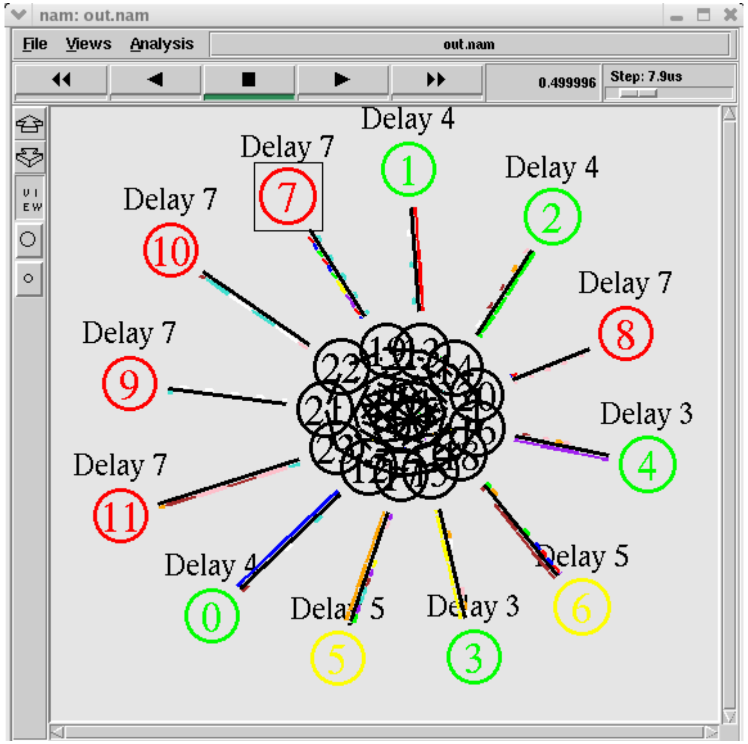
Implementation in ns-2

- Code model used to connect peers, blue circles represent game applications, red their TCP agents, and yellow the wrapper TCPApps



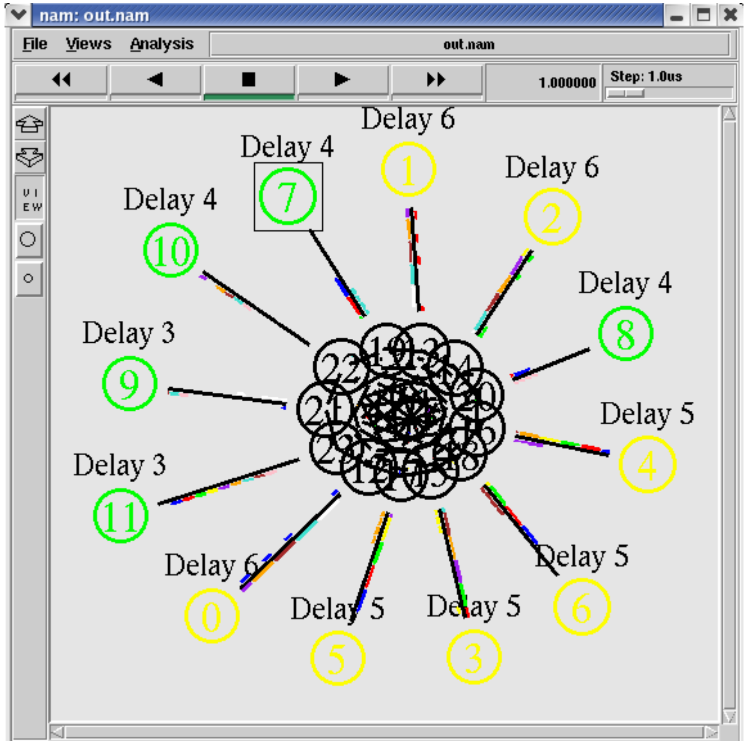
Implementation in ns-2

- Sample nam output (startup phase)



Implementation in ns-2

- Sample nam output (steady state)



Discussion of Results

- Initial Results
  - First run used ADSL connections, 384 kbps upstream, using 25 frames per second with 64 bytes constant data packet size
  - Up to 13 peers could participate
  - Bandwidth required for 13 nodes = 12 remote peers * 25 fps * 104 bytes = 250kbps (or 65% of available bandwidth)

List of Remaining Work

- Investigate asymmetric networks
  - Cable, ADSL, and possibly modems
- Real-world conditions
  - Varying traffic models
  - Lossy connections
  - Interrupting competing traffic
- Improve sequencing of packets

List of Remaining Work

- Enhancements other could add to this project:
  - Compare to UDP
  - Compare different flavours of TCP
  - Use real traffic traces
  - Smart algorithms

Conclusion

- So far, so good :o)
- Network was successfully implemented, and simulated
- Even a simple model successfully connected 13 peers
- Lots of fun work ahead

Conclusion

- References

1. Postel, J. “TRANSMISSION CONTROL PROTOCOL”, IETF-RFC-793, http://www.ietf.org/rfc/rfc0793.txt?number=793, September 1981
2. Bonham, S., Grossman, D., Portnoy, W., Tam, K. “Quake: An Example Multi-User Network Application – Problems and Solutions in Distributed Interactive Simulations”, University of Washington, http://www.cs.washington.edu/homes/grossman/projects/561projects/quake/Quake.ps, May 2000
3. Bettner, B., Terrano, M. “1500 Archers on a 28.8: Network Programming in Age of Empires and Beyond”, http://www.ensemblestudios.com/openjournal4/story/networking.ppt, March 2001
4. Borella, M. “Source Models of Network Game Traffic”, 3Com Corp., http://www.borella.net/mike/academics/game-traffic.pdf, November 2003
5. Faerber, J. “Network Game Traffic Modelling”, Institute of Communication Networks and Computer Engineering, http://www.ibr.cs.tu-bs.de/events/netgames2002/presentations/faerber.pdf, April 2002

Conclusion

- Thank you!
- Questions?